COMMODORE LOGO

ACTIVITIES FOR EXPLORING TURTLE GRAPHICS

H.J. Bailey • T.H. Doran • K.M. Brautigam

Bailey
Doran
Brautigam

COMMODORE LOGO

ACTIVITIES FOR EXPLORING
TURTLE GRAPHICS

Brady

Commodore Logo

Publishing Director: David Culverwell
Acquisitions Editor: Gisele Asher
Production Editor: Roberta Glencer
Text Design: Michael J. Rogers
Art Director/Cover Design: Don Sellers
Assistant Art Director: Bernard Vervin
Photographer: Leslie McMahon
Manufacturing Director: John Komsa

Typesetting: Compolith Graphics, Indianapolis, IN
Printing: Fairfield Graphics, Fairfield, PA
Typefaces: Helvetica (display heads), Paladium (text), and OCR-B (Logo programs)

Commodore Logo

Activities for Exploring Turtle Graphics

Harold J. Bailey
Bloomsburg University
Bloomsburg, Pennsylvania

Kathleen M. Brautigam
Milton Area School District
Milton, Pennsylvania

Trudy H. Doran
Shikellamy School District
Sunbury, Pennsylvania

Brady Communications Company, Inc.
A Prentice-Hall Publishing Company

Bowie, MD

Commodore Logo: Activities For Exploring Turtle Graphics

Copyright © 1984 by Brady Communications Company, Inc. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or by any information storage and retrieval system, without permission in writing from the publisher. For information, address Brady Communications Company, Inc., Bowie, Maryland 20715.

Library of Congress Cataloging in Publication Data
Bailey, Harold J., 1940- Commodore logo.
Includes index.
Summary: Activities for using the computer language LOGO for graphics as creative expression.
1. Computer graphics—Juvenile literature.
2. LOGO (Computer program language)—Juvenile literature.
3. Commodore 64 (Computer)—Programming—Juvenile literature. [1. Computer graphics. 2. LOGO (Computer program language) 3. Commodore 64 (Computer)—Programming] I. Brautigam, Kathleen M. II. Doran, Trudy H. III. Title.
T385.B345 1984 001.64'43 84-12422
ISBN 0-89303-376-6

Prentice-Hall International, Inc., London
Prentice-Hall Canada, Inc., Scarborough, Ontario
Prentice-Hall of Australia, Pty., Ltd., Sydney
Prentice-Hall of India Private Limited, New Delhi
Prentice-Hall of Japan, Inc., Tokyo
Prentice-Hall of Southeast Asia Pte. Ltd., Singapore
Whitehall Books, Limited, Petone, New Zealand
Editora Prentice-Hall Do Brasil LTDA., Rio de Janeiro

Printed in the United State of America

84 85 86 87 88 89 90 91 92 93 94 1 2 3 4 5 6 7 8 9 10

Dedications

To Brad, for being my best friend.
Kathy

To my parents, for years of support, love, and encouragement.
Trudy

To my family, for their continual patience and understanding during a period when my professional aspirations have affected our moments together.
Hank

Table of Contents

Preface ix

Chapter 1 Exploring Turtle Territories 1
- Activity 1.1 Turtle Moves 1
- Activity 1.2 Turtle Pivots 7
- Activity 1.3 Turtle Toggles 14
- Activity 1.4 Turtle Repetitions 19
- Activity 1.5 Turtle Circles 25
- Activity 1.6 Turtle Arcs 32
- Reference Point 1 36

Chapter 2 Preserving Turtle Tracks 41
- Activity 2.1 Turtle Procedure 41
- Activity 2.2 Turtle Files 50
- Activity 2.3 Turtle Inventory 58
- Activity 2.4 Turtle Editing 64
- Activity 2.5 Turtle Controls 76
- Activity 2.6 Turtle Maintenance 83
- Reference Point 2 93

Chapter 3 Extending Turtle Trips 97
- Activity 3.1 Turtle Input 97
- Activity 3.2 Turtle Hues 109
- Activity 3.3 Turtle Background 114
- Activity 3.4 Turtle Variables 125
- Activity 3.5 Turtle Iterations 134
- Activity 3.6 Turtle Limits 144
- Activity 3.7 Turtle Conditions 158
- Activity 3.8 Turtle Tricks 172
- Reference Point 3 180

Chapter 4 Coordinating Turtle Points 189
- Activity 4.1 Turtle Positions 189
- Activity 4.2 Turtle Placement 200
- Activity 4.3 Turtle Tessellations 207
- Reference Point 4 216

Chapter 5 Recruiting the Turtle's Team 225
Activity 5.1 Sprinting Sprites 225
Activity 5.2 Springing Sprites 232
Activity 5.3 Scaling Sprites 241
Activity 5.4 Shaping Sprites 246
Reference Point 5 255

Chapter 6 Creating Turtle Temptations 263
Introduction 263
Part-to-Whole Programming 264
Whole-to-Part Programming 271
Concluding Remarks 282
Suggested Projects 282

Resource List 283

Selected Answers To Turtle Testers 287

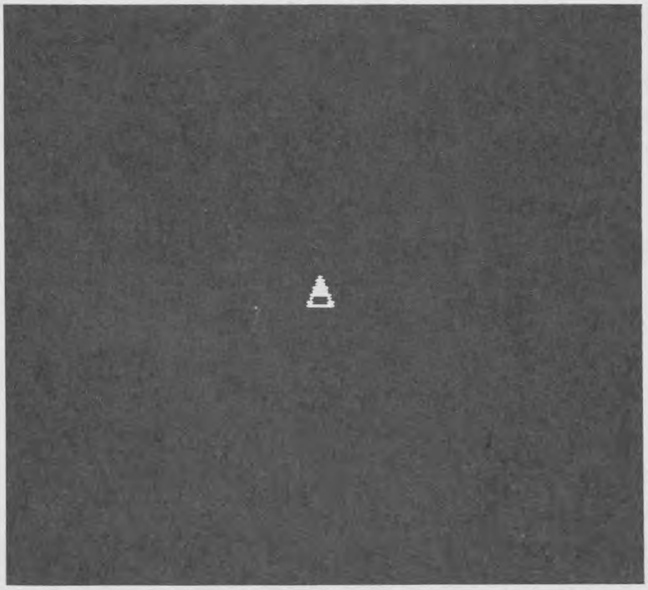
Index 301

Preface

Who enjoys drawing more than children? Why are art classes usually included in the public school curriculum? It is because youngsters find the drawing of pictures an enjoyable and effective way of expressing their ideas and thoughts.

Traditionally, children have used pencil, crayon, and paint to demonstrate their artistic talents. The microcomputer makes a new world of graphics and color available to everyone.

One of the computer languages with graphic capabilities is Logo. Logo represents a philosophy of education advocated by Seymour Papert at the Massachusetts Institute of Technology. Through Logo, Papert maintains that children gain control over the learning process by teaching the computer, rather than being taught by it. Logo provides an environment where the computer becomes a tool to facilitate learning. A child first explores and plays with an idea and observes the results immediately through the path of a "turtle" illustrated in the diagram below.



The youngster discovers relationships among several ideas and formalizes them by creating a procedure. The development of procedures becomes a personal repertoire of building blocks through which logical thinking, problem solving skills, and creativity are enriched.

There are versions of Logo for virtually every popular home computer. The purpose of this book is to demonstrate how Commodore Logo uses Turtle Graphics to facilitate the application of Papert's theory. While Commodore Logo also has arithmetic operations and list-processing capabilities, this text concentrates solely on Turtle Graphics.

To use this book, you must have a Commodore 64 system. You also need one 1541 Disk Drive to use your Commodore Logo diskette. Access to a color monitor or a color TV is desirable to fully appreciate the program examples in the text.

This book consists of a collection of instructional activities that progress in a systematic manner, culminating in a comprehensive understanding of Turtle Graphics. No previous computer experience is required. All Logo commands, geometric concepts, and elementary computer jargon are defined when introduced and are reinforced throughout the book.

The selection and sequence of activities within each chapter are based on the authors' teaching experience and interaction with children exploring Logo. Integrating this practical knowledge of an appropriate learning sequence, *Commodore Logo: Activities For Exploring Turtle Graphics* provides a more direct and accelerated approach for the following audiences.

Parents: offers an opportunity for parents and children to explore Logo together

Educators: provides an instructional guide for introducing Logo to children

Prospective educators: serves as a textbook for a college/university level Logo course

Independent learners: allows for individualized study of Logo for the able student and motivated hobbyist aged 12 and up

Each activity includes five sections.

Turtle Trials
Turtle Talk
Turtle Truth
Turtle Tester
Turtle Teaser

The **Turtle Trials** section presents a series of "hand-on" activities where the reader sits at a microcomputer and enters given Logo commands. The reader is asked to describe what is observed in the resulting displays. These results demonstrate both the proper usage and common errors associated with the activity objective. You are encouraged to write descriptions, notes, and observations in the book.

The **Turtle Talk** section provides narrative reinforcement for the interactive "hands-on" experience.

The **Turtle Truth** section provides a comprehensive example that demonstrates a correct application of the newly introduced commands and presents one of the many possible graphic designs using Turtle Graphics.

The **Turtle Tester** section requests the reader to demonstrate an understanding of the activity objective by writing a Logo program for a given situation.

Finally, the **Turtle Teaser** section suggests an activity of an extended nature to further explore, refine, and integrate Logo skills.

The Logo activities form six chapters that should be followed sequentially. Chapter 1 shows how one controls the movement of the "turtle" through simple Logo commands. In Chapter 2, these commands and others are combined to form procedures. The reader also becomes skillful in using the Logo editor. Chapter 3 introduces color and the concept of recursion, to extend the development of procedures to include variables. In Chapter 4, coordinates are used as a means by which graphic designs can be displayed and animated in an effective manner. Chapter 5 deals with sprites. Finally, Chapter 6 presents several different approaches by which complex screen displays can be designed.

Logo is designed to provide a medium for meaningful and enjoyable learning experiences. The authors anticipate that *Commodore Logo: Activities For Exploring Turtle Graphics* will guide the reader toward a proficiency in the use of Turtle Graphics and provided an opportunity for creative expression.

H.J.B.
K.M.B.
T.H.D.

Limits of Liability and Disclaimer of Warranty

The authors and publisher of this book have used their best efforts in preparing this book and programs contained in it. These efforts include the development, research, and testing of the programs to determine their effectiveness. The authors and publisher make no warranty of any kind, expressed or implied, with regard to these programs, the text, or the documentation contained in this book. The authors and publisher shall not be liable in any event for claims of incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of the text or the programs. The programs contained in this book are intended for the use of the original purchaser.

Note to Authors

Do you have a manuscript or a software program related to personal computers? Do you have an idea for developing such a project? If so, we would like to hear from you. The Brady Company produces a complete range of books and applications software for the personal computer market. We invite you to write to David Culverwell, Publishing Director, Brady Communications Company, Inc., Bowie, MD 20715.

Trademarks of Material Mentioned in This Text

Commodore 64 is a registered trademark of Commodore Business Machines, Inc.

Commodore Logo is a trademark of Logo Computer Systems, Inc.

Acknowledgments

Special thanks are extended to Irene Fischer for her help in reviewing, testing, and critiquing our manuscript.

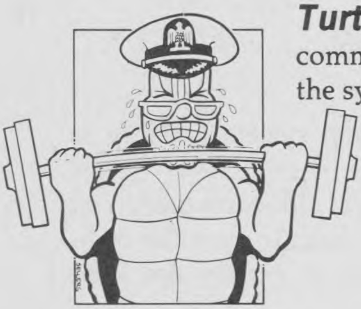
We express our gratitude to each family and many friends for their patience and constant support during the sometimes hectic period of writing this book.

Brad, Kristin, and Phillip
David, Darci, and Tonya
Linda, Kimberly, and Keith
and many encouraging friends

Exploring Turtle Territories

1

Activity 1.1 Turtle Moves



Turtle Trials. This activity introduces fundamental commands that control the appearance and movement of the symbolic "turtle" used in Logo.

1. Turn on the disk drive. The switch is located on the right portion of the rear panel. Now turn on your monitor or TV. Finally, turn on the Commodore 64 computer system. You will receive the following message:

```
         **** COMMODORE 64 BASIC V2 ****

      64K RAM SYSTEM 38911 BASIC BYTES FREE

READY.

■
```

The blinking box is called the "cursor" and indicates that the computer is ready to accept instructions at that location. These instructions or commands are commonly known as "input."

Open your disk drive door, insert the Commodore Logo diskette, close the disk drive door, and type

```
"LOAD LOGO", 8
```

Now press the RETURN key. The red light on the disk drive will go on and the following messages will appear.

```
SEARCHING FOR LOGO
LOADING
READY.
```

Type RUN and press the RETURN key. The disk drive will activate and you will receive the following message.

```
LOADING, PLEASE WAIT . . .
```

After a few minutes you will receive another message.

```
                     COMMODORE 64 LOGO

COPYRIGHT (C) 1982, 1983 TERRAPIN, INC.
COPYRIGHT (C) 1981 MIT

WELCOME TO LOGO!
?■
```

The ? is a "prompt" that indicates the computer system is ready to accept a Logo command.

This entire process is known as "cold booting" the computer. You will need to cold boot the system each time you wish to turn the computer on and work with Logo. Follow the steps in exactly the same order as given above.

NOTE Please follow the loading procedure described above for all of the activities in Chapter 1. It will be assumed that you have completed this "booting" process prior to each activity.

Type the word DRAW and press the key marked RETURN. Describe the resulting screen display on the line provided.

DRAW ____________________________________

NOTE In all of the following activities you will be presented with directions and then a list of commands to type. Please assume that after typing the command you should press the RETURN key and observe the screen display. If a typing error occurs, it can be corrected by backspacing with the INST DEL key (located in the upper right corner of the keyboard). Each time the INST DEL key is pressed, the cursor moves one position to the left, erasing the character it covers. This process is repeated until the undesired characters are removed. Then retype the correct set of characters.

If you fail to recognize an error and press RETURN, retype the entire line.

2. Enter the following sequence of commands and describe what effect each command has on the screen display.

```
FORWARD 40 ________________________________
FORWARD 10 ________________________________
FORWARD20 _________________________________
FD 30 _____________________________________
FD ________________________________________
DRAW ______________________________________
HIDETURTLE ________________________________
BACK 25 ___________________________________
SHOW TURTLE _______________________________
SHOWTURTLE ________________________________
BK 10 _____________________________________
HT ________________________________________
BK20 ______________________________________
BK 15 _____________________________________
ST ________________________________________
BK 5 ______________________________________
BK ________________________________________
DRAW ______________________________________
```

3. Enter the following commands and observe the screen display.

```
FD 40
FD 40
FD 40
FD 5
FD 5
FD 5
FD 60
```

Where is the turtle (wedge-like object) originally located on the screen?

____

Starting from that original position, what is the total number of forward (FD) moves before the turtle disappears from the screen? ________

Describe where the turtle reappears. ______________________

4. Enter the following sequence of commands and observe the screen display.

```
DRAW
BK 30
BK 30
BK 20
BK 3
BK 1
BK 60
```

Where is the turtle originally located on the screen? ________________

Starting from that original position, what is the total number of moves backward (BK) before the turtle leaves the gray portion of the screen?

____

Describe where the turtle reappears. ________________________

5. Enter the following commands. Observe and describe the effect on the screen display.

```
DRAW
FD 135 ____________________________________
FULLSCREEN ________________________________
```

NOTE While in FULLSCREEN, you are not able to view what is typed. Type carefully!

```
TEXTSCREEN ________________________________
SPLITSCREEN _______________________________
```

6. There are function keys marked f# (where # = 1, 2, 3, . . . , or 8) on the right side of the keyboard. These keys are used as alternatives to the FULLSCREEN, TEXTSCREEN, and SPLITSCREEN commands. To experiment, enter the following commands.

```
DRAW
FD 130
FD 130
f1 ________________________________________
f5 ________________________________________
f3 ________________________________________
```

What is the vertical length of the screen when fullscreen display is in effect? ________________________________

What full word command is equivalent to f1 ________________

What full word command is equivalent to f5? ________________

What full word command is equivalent to f3? ________________

7. Enter the following sequence of commands and observe the screen display. Describe what happens after each section.

```
DRAW
FD 60
FD -60 ________________________________
FD 70
BK 70 _________________________________
BK 40
BK -40 ________________________________
```

Turtle Talk. Part 1 demonstrates that access to Turtle Graphics is obtained by cold booting with the Logo diskette.

In Part 2, DRAW causes a small wedge-like object known as the turtle to appear on the screen. This turtle simulates the communication between you and drawings produced on the display screen. The appearance and disappearance of the turtle in FULLSCREEN or SPLITSCREEN mode is controlled by the commands SHOWTURTLE (abbreviated ST) and HIDETURTLE (abbreviated HT). HT and ST appear to have no effect in TEXTSCREEN mode. If a space is inserted between the words of these commands, an error message is given. DRAW is used to erase all drawings and place the turtle in the center of the screen pointing up.

FORWARD (abbreviated FD) and BACK (abbreviated BK) are movement commands introduced in Parts 3 and 4. FD moves the turtle in the direction in which it is pointing. BK moves the turtle in the opposite direction. The length of the path taken by the turtle in "turtle units" is defined by the number following the FD or BK command. If no number follows the command, an error message such as "FD (or BK) NEEDS INPUTS" occurs. The command must then be re-entered properly.

Part 7 demonstrates that it is possible to enter a negative value as input following a FD or BK command. This causes the turtle to move in the reverse direction of the command used. The incorporation of negative values is important when demonstrating certain mathematical principles. Note that while negative inputs are permitted, only positive values are needed since both the FD and BK commands are available for providing turtle movement in opposite directions.

Parts 5 and 6 show there are three methods of viewing the screen. SPLITSCREEN (f3) is the normal way of viewing the screen. This allows you to view graphics (colored in gray) and also see five lines of text (colored in blue). The screen dimensions in this display mode are 130 units forward

(FD) from the turtle's starting position and 84 units back (BK) from the turtle's starting position. The difference between 130 and 84 is space used for text.

FULLSCREEN (f5) lets you see the total graphics screen which is 260 units high or 130 units forward (FD) from the turtle's starting position and 130 units back (BK) from the turtle's starting position. This display temporarily removes the five lines of text which appear at the bottom of the screen, commonly called the "text window." This lets you see if the turtle is drawing in the section under the text window. If a command is given that sends the turtle off the screen, it wraps around to reappear and continue drawing at the opposite edge of the screen.

TEXTSCREEN (f1) lets you view the commands that have been previously entered, but have scrolled off from view in the text window. The display screen commands do not erase the existing drawing or text.

Turtle Truth. Study the following commands to determine the screen display. Check your answers by entering the commands.

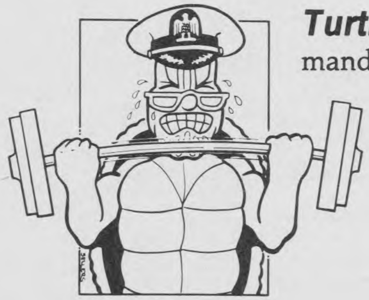
```
DRAW
BK 50
HT
FD 80
ST
FD 40
FD 100
f5
DRAW
f1
f3
HT
FD 40
ST
FD -100
HT
DRAW
```

Turtle Tester. Write a sequence of commands that perform the following operations in this order. (Answer on page 287.)

A. Use DRAW to clear the screen.
B. Hide the turtle and move it forward 120 units.
C. Show the turtle and move it back 75 units.

Turtle Teaser. Use no more than four Logo commands to draw a vertical line from the top of the screen to the bottom of the screen in FULLSCREEN mode.

Activity 1.2 Turtle Pivots



Turtle Trials. This activity introduces the use of commands to set the heading of the turtle.

NOTE HEADING is the term used to denote the direction of the turtle. When the turtle first appears on the screen, the starting position of the tip will always be upward or northerly.

1. Enter the following sequence of commands and notice the change in

HEADING shown by the turtle. Describe what happens after each command.

```
DRAW
RT 90
FD 25
DRAW
RT
RT 180
FD 50
DRAW
RT 270
FD 75
DRAW
RT360
RT 360
FD 100
```

2. Enter the following sequence of commands and notice the change in HEADING shown by the turtle. Describe what happens after each command.

```
DRAW
LT 90
FD 50
BK 50
LT 90
FD 50
BK 50
LT 90
FD 50
BK 50
LT 90
FD 50
BK 50
```

NOTE An angle is a figure formed by two lines extending from a common point. The size or measure of an angle can be expressed in degrees, where a degree represents a 1/360 part of the circumference (or perimeter) of a circle. Figure 1-1 illustrates a 45 degree angle. The LT and RT commands change the direction of the turtle, permitting it to revolve in an angular fashion.

How many degrees does the turtle revolve with LT 90?

How many degrees does the turtle revolve with RT 270?

How many degrees does the turtle revolve when it completes one revolution by returning to its starting position?

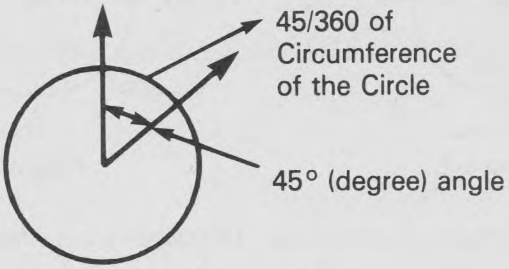


Figure 1-1.

3. Enter the following commands and observe the heading of the turtle after each section.

```
DRAW
RT 45
LT 45
```

Describe what happens.

```
RT 180
RT 180
```

Describe what happens.

```
LT 90
LT 180
```

Describe what happens.

```
LT 15
LT 50
LT 25
```

Describe what happens.

What is the sum of the last three input numbers?

Does the turtle draw when using an LT or RT command?

What does it do?

What commands must you use to make it draw?

4. Figures 1-2 and 1-3 display the turtle in its starting position. Imagine the

turtle making RIGHT and LEFT turns. Fill in the numerical values that indicate the missing degrees of revolution for each of the lettered positions.

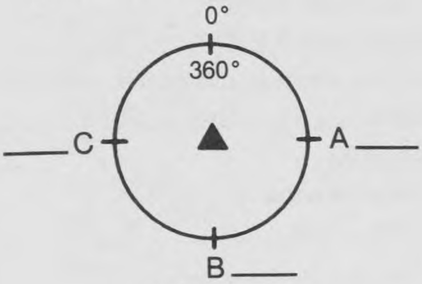


Figure 1-2.

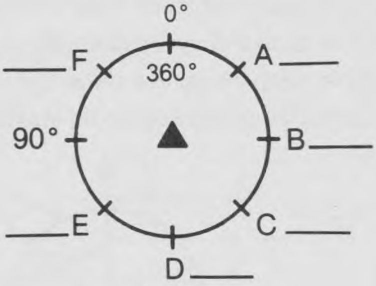


Figure 1-3.

5. Enter the following commands. Describe what happens after each command.

NOTE SETH represents SETHeading.

```
DRAW
SETH 90  ________________________________
FD 100   ________________________________
SETH 0   ________________________________
FD 30    ________________________________
SETH 270 ________________________________
FD 100   ________________________________
SETH 180 ________________________________
FD 30    ________________________________
SETH 360 ________________________________
```

Describe the shape that is drawn. ____________________

Are the SETH commands based on RT or LT commands? ________

Why is SETH 360 the last command? ____________________

What is another command that could have been used instead of SETH 360? ____________________

6. Enter the following commands.

NOTE PR represents PRint.

```
DRAW
RT 97
```

```
LT 42
RT 405
PR HEADING
```

What reply does the computer give you to the command PR HEADING? __________

What two commands can you type to return the turtle to its starting position from its current heading? RT __________ or LT __________

7. Enter these commands again, but add HOME at the end.

```
DRAW
RT 97
LT 42
RT 405
PR HEADING
HOME
```

What other commands can be used to return the turtle to its starting position? LT __________ or RT __________

8. Enter the following commands and observe the screen display.

```
DRAW
RT 105
FD 40
LT 37
FD 90
HOME
```

Describe the position of the turtle. __________

What geometric figure is drawn? __________

Turtle Talk. The turtle is represented by a triangular object with a solid tip. The direction this tip is facing is called the HEADING. The commands RT and LT, introduced in Parts 1, 2, and 3, use an input value to determine the number of degrees the turtle turns. The RT command provides clockwise revolution while the LT command provides counterclockwise revolution. If the commands RT or LT are entered without an input value, the error message "RT (OR LT) NEEDS MORE INPUTS" will appear. Notice that space remains an important part of the command. Changing the heading of the turtle does not make it draw. Only FD or BK followed by a number allows the turtle to draw. By using these four commands in sequence, you are able to draw in any direction. A complete revolution of the turtle is 360 degrees. It will turn right or left from any starting point.

Part 5 shows you can use the SETH (SETHeading) command with an input number to point the turtle to a predefined direction. The SETH command heading is determined by the clockwise revolution of the turtle in relation to its starting position. In contrast, the RT and LT commands turn the turtle in relation to its present position. The commands HOME, SETH 0, and SETH 360 appear to have similar control over the heading of the turtle. Each command turns the turtle in a northern direction. However, Part 8 demonstrates the HOME command will also send the turtle to the center of the screen and draw a path along the way.

If you get confused concerning the heading of the turtle, you can ask the computer to print the present heading. The command PR HEADING in Parts 6 and 7 causes the computer to print the heading on the screen. The PR HEADING output gives the numerical value of the current clockwise direction relative to the HOME position. In Part 5 the Logo commands were terminated with SETH 360 or HOME. It is good practice to return the turtle to its starting (HOME) position after each drawing so that the status of the turtle is known for future drawing.

Turtle Truth. Study the following Logo procedure to determine the screen display. Check your answer by entering the commands.

```
DRAW
RT 90
FD 100
LT 90
FD 50
SETH 270
FD 200
LT 90
FD 50
SETH 90
FD 200
BK 100
HOME
PR HEADING
```

Turtle Tester. Draw three rectangles of different sizes. Return the turtle to its starting position after the drawing is completed. (Answers on page 287.)

A. Draw a rectangle using only RT commands for the corners. Make the rectangle tall.
B. Draw a rectangle using only LT commands for the corners. Make the rectangle wide.
C. Draw a rectangle any way you want, but use only SETH commands for the corners.

Turtle Teaser. Draw the rectangle illustrated in Figure 1-4.

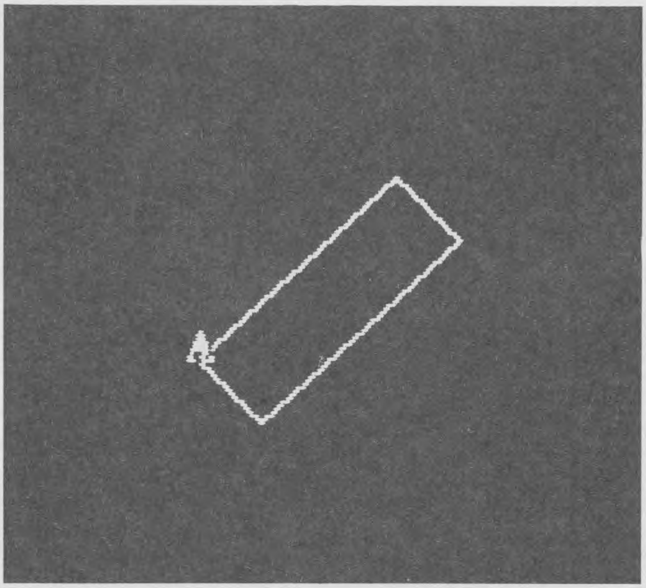
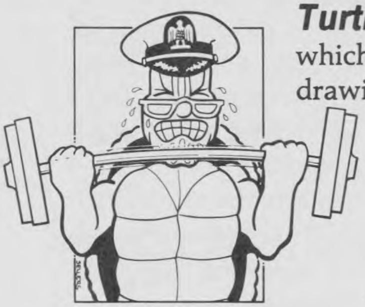


Figure 1-4.

Activity 1.3 Turtle Toggles



Turtle Trials. This activity introduces the commands which permit the user to relocate the turtle and alter its drawing mode.

1. Enter the following commands.

```
DRAW
LT 180
PU
FD 40
```

NOTE PU is an abbreviation for PENUP.

What movement do you observe? ______________________

2. Add the following commands and describe the effect each command has on the screen display.

```
RT 90 ____________________________
FD 40 ____________________________
RT 90 ____________________________
```

Does any drawing take place? ______________________

3. Add the following commands.

```
PD
FD 80
```

NOTE PD is an abbreviation for PENDOWN.

What do you observe? ______________________

4. Add the following commands.

```
RT 90
FD 40
```

```
RT 90
FD 80
RT 90
FD 40
```

What geometric shape is drawn? ____________________

What is the heading? ____________________

Type the command

```
PR HEADING
```

What is the computer's response? ____________________

5. Type the following commands and describe the effect each command has on the screen display.

`PENERASE` ____________________

Does the turtle change its appearance? ____________________

`BK 40` ____________________

`FD 40` ____________________

`BK 40` ____________________

`PENCOLOR 1` ____________________

Does the turtle change its appearance? ____________________

`FD 40` ____________________

6. Add the following commands.

`PENCOLOR -1` ____________________

Does the turtle change its appearance? ____________________

`BK 40` ____________________

`PC 1` ____________________

Does the turtle change its appearance? ____________________

`FD 40` ____________________

`PC -1` ____________________

`BK 40` ____________________

`PC 1` ____________________

What commands place the turtle in the erasing mode? __________ or __________

```
CS
```

Describe what happens. ______________________________

7. Enter the commands and observe what takes place after each section.

```
HOME
DRAW
LT 90
FD 50
RT 90
FD 90
BK 180
DRAW
```

Describe what happens. ______________________________

```
LT 90
FD 25
RT 90
FD 50
BK 100
HOME
```

Describe what happens. ______________________________

```
DRAW
LT 90
FD 25
RT 90
FD 50
BK 100
PU
HOME
PD
```

Describe what happens. ______________________________

```
RT 90
FD 25
LT 90
FD 50
BK 100
PU HOME PD
HT
```

Describe what happens. ______________________________

Turtle Talk. PENUP, PENDOWN, and PENERASE are additional commands that control the drawing mode in Turtle Graphics. The turtle's PEN is always DOWN and ready to draw lines when the turtle enters the screen.

Parts 1, 2, and 3 demonstrate that when PENUP (abbreviated PU) is entered, the turtle's pen is no longer in position to draw. It allows relocation of the turtle on the screen without drawing. Once the PENUP (abbreviated PU) command is issued, there is no drawing until the order is reversed by entering PENDOWN (abbreviated PD).

In Part 5, PENERASE or PENCOLOR −1 (abbreviated PC −1) changes the drawing mode so that any line or dot in the path of the turtle is erased. It remains in erase mode until PENCOLOR 1 (abbreviated PC 1) is entered. The appearance of the turtle indicates its current mode. A common error when using PU is forgetting to return to the PD mode. Similarly, a common error in using PENERASE or PC −1 is forgetting to return to a drawing state with a PC 1 command.

CLEARSCREEN, DRAW, and HOME are other useful commands. Although similar, each performs a distinct function. CLEARSCREEN (abbreviated CS) erases the drawing on the screen but does not affect the position and heading of the turtle. DRAW erases all drawing and also returns the turtle to the HOME position.

When the command HOME is entered, the turtle returns to its HOME position, drawing a path and leaving all previously drawn lines on the screen. To avoid leaving an unwanted path, precede the command HOME with PU and follow it with PD. Although it is not necessary to type these commands on one line, it helps to visualize them as a unit performing one function. The recommended format to complete a drawing is to conclude with this series of commands. It provides uniformity and may prevent errors.

Turtle Truth. Study the following commands and predict what and where the turtle is to draw. Check your prediction by entering the commands.

```
DRAW
FD 30
```

```
CS
LT 45
FD 70
PENERASE
BK 70
PC 1
SETH 0
FD 80
RT 90
FD 40
RT 90
FD 80
RT 90
FD 40
PC -1
BK 40
RT 90
PC 1
FD 40
LT 90
FD 40
PU HOME PD
HT
```

Turtle Tester. Draw the letter B as shown in Figure 1-5. (Answer on page 288.)

Turtle Teaser. Draw your initials on the screen in an interesting arrangement.

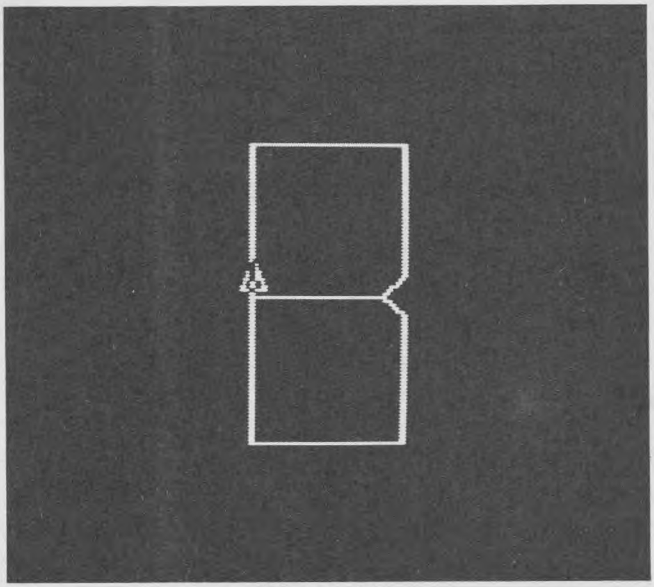
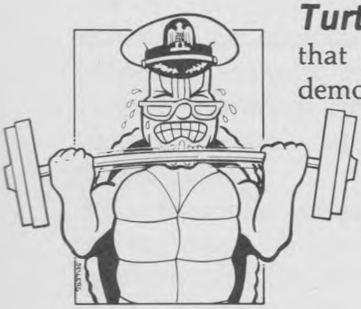


Figure 1-5.

Activity 1.4 Turtle Repetitions



Turtle Trials. This activity introduces a technique that simplifies the writing of repetitive commands and demonstrates how to draw regular polygons.

1. Enter the following commands and comment on each section.

```
DRAW
FD 50
RT 90
FD 50
RT 90
FD 50
RT 90
```

```
FD 50
RT 90
```

What diagram is drawn on the screen? ______________________

What direction is the heading? ______________________

What two commands are repeated? ______________________

How many times are those commands repeated? ______________________

2. On the Commodore 64, the [bracket is obtained by pressing the SHIFT key and the : key at the same time. The] bracket is obtained by pressing the SHIFT key and the ; key. These brackets must be used with the REPEAT command. Enter the following command.

```
REPEAT 4 [FD 50 RT 90]
```

What diagram is on the screen? ______________________

Count the corners. ______________________

What is the product of the number of corners and the size of one angle (4*90)? ______________________

NOTE The symbol for multiplication is *. For example, to multiply 5 by 8, type 5*8.

3. Enter the following command and observe the screen display.

```
PENERASE
REPEAT 4 [FD 50 RT 360/4]
```

Describe what happens. ______________________

NOTE The symbol for division is /. For example, to divide 12 by 3, type 12/3.

4. Enter the following commands and describe the effect of each command.

```
PC 1 ______________________
REPEAT 3 [FD 80 LT 360/3] ______________________
PR 360/3 ______________________
PC -1 ______________________
REPEAT 3 [FD 80 LT 120] ______________________
PC 1 ______________________
```

5. Enter the following commands and describe what happens.

```
REPEAT 6 [FD 50 RT 360/6] ______________________
```

```
PR 360/6 ________________________________
FD 50 ___________________________________
```

6. Enter these commands and record the effect each one has on the screen display.

```
DRAW
REPEAT 7 [FD 50 LT 360/7] _______________
PR 360/7 ________________________________
PR 360/8 ________________________________
DRAW
REPEAT 8 [FD 40 LT 45] __________________
```

Turtle Talk. A polygon is a closed figure consisting of three or more connected line segments. Polygons are classified by their number of sides. Names of polygons with three to ten sides are as follows: triangle (three sides), quadrilateral (four sides), pentagon (five sides), hexagon (six sides), heptagon (seven sides), octagon (eight sides), nonagon (nine sides), and decagon (ten sides).

A geometry theorem states that the sum of the exterior angles of any polygon is always 360 degrees. Figure 1-6 illustrates that an exterior angle (that is angle A) is formed by extending a side of a polygon (dotted line) and considering the angle formed by that side and the adjacent side. Other exterior angles (angles B,C,D,E) are formed in a similar manner by extending each of the sides in succession. There are as many exterior angles for a polygon as the number of sides. In contrast to exterior angles, interior angles are the remaining angles "inside" the polygon.

A polygon is regular if it is equilateral (all sides the same length) and equiangular (all angles the same measure). If the polygon is regular, it can be shown that the exterior angles are equal. Thus, in Figure 1-6, the measure of angle A is 72 degrees (360/5), since the regular pentagon has five sides. Figure 1-7 displays some other common regular polygons and their corresponding exterior angle measures. Notice the square is the only regular polygon where the interior and exterior angles have the same measure.

Parts 2, 3, and 4 demonstrate that the turtle forms a polygon by completing an entire 360 degree rotation. Since the turtle's revolution created by the RT and LT commands corresponds to the exterior angles of the polygon, dividing 360 by the number of sides of the polygon determines the appropriate values for RT or LT. Equivalently, the product of the number of angles times the size of one angle will equal 360. Seymour Papert refers to this as the "TOTAL TURTLE TRIP THEOREM." This principle is useful when drawing any regular polygon. It allows the user to rely on the computer to calculate the size of exterior angles by the quotient 360/N, where N is the number of sides of the regular polygon.

By using a combination of this principle and the REPEAT command, the user has an efficient method of drawing regular polygons. The REPEAT command provides a convenient way to combine groups of statements that occur more than once. This command requires a numerical value before the square brackets ([]) to indicate the number of times the sequence within the brackets is to be performed. For example, regular polygons are drawn with the following single line commands.

```
REPEAT 4 [FD 50 RT 360/4] (square)
REPEAT 3 [FD 80 LT 120] (triangle)
REPEAT 6 [FD 50 RT 60] (hexagon)
REPEAT 7 [FD 50 LT 360/7] (heptagon)
```

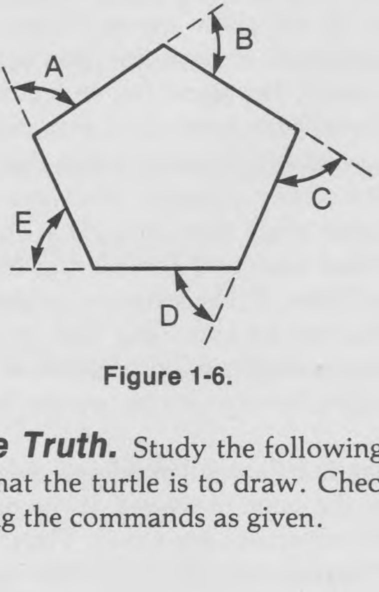


Figure 1-6.

Turtle Truth. Study the following commands and predict what the turtle is to draw. Check your prediction by entering the commands as given.

```
DRAW
PU
FD 30
PD
```

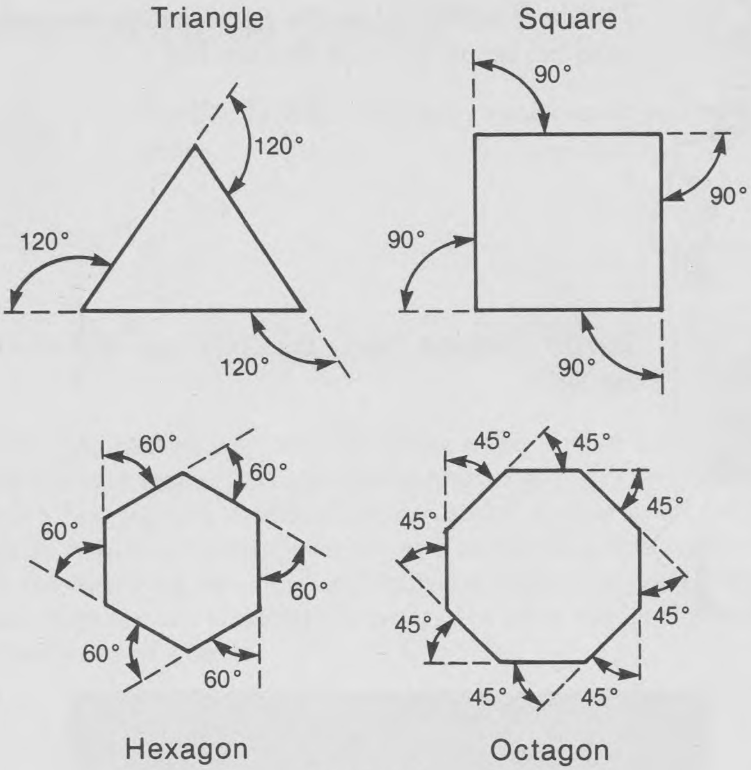


Figure 1-7.

```
REPEAT 8 [FD 40 LT 360/8]
REPEAT 8 [FD 40 RT 360/8]
PR 360/8
RT 135
FD 40
RT 90
FD 40
SETH 0
BK 40
REPEAT 8 [FD 40 LT 45]
REPEAT 8 [FD 40 RT 45]
PU HOME PD
f5
f1
f3
DRAW
```

Turtle Tester. Make the same arrangement of shapes using hexagons. (Answer on page 288.)

Turtle Teaser. Make the STOP sign illustrated in Figure 1-8.

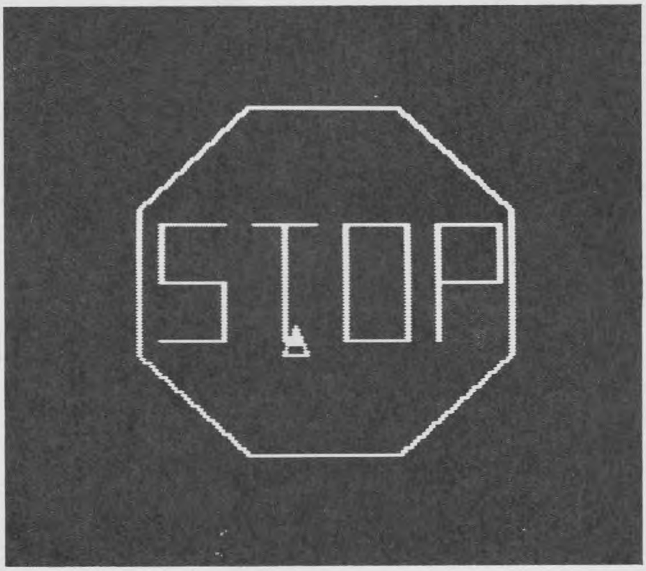
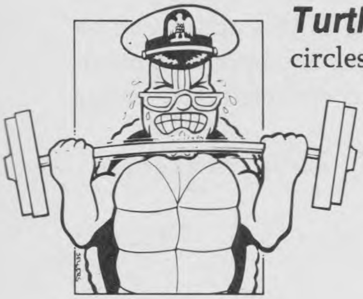


Figure 1-8.

Activity 1.5 Turtle Circles



Turtle Trials. This activity demonstrates how to draw circles.

NOTE Mathematically, a circle is the collection of all points that are the same distance from a common point known as the center. The radius of a circle is a line segment drawn from the center to some point on the circle. A diameter, which is twice the length of a radius, is a line segment passing through the center of the circle and touching two of its points. There is an unlimited number of radii and diameters for each circle. Figure 1-9 illustrates these parts of a circle.

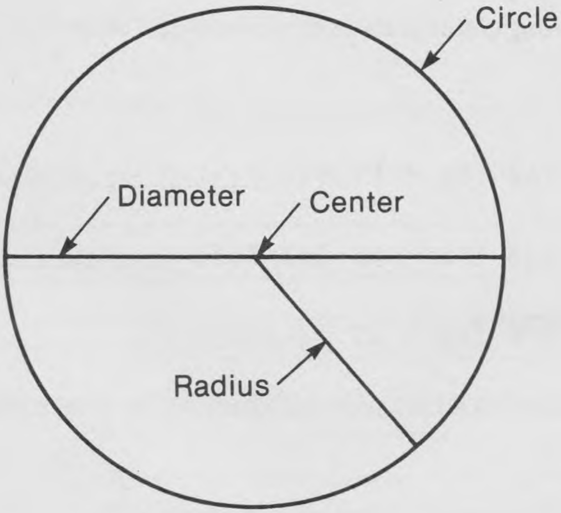


Figure 1-9.

1. Cold boot the system with the Commodore Logo diskette. When you receive the WELCOME TO LOGO! message, remove the Logo diskette from the drive, insert the Logo Utility Disk, type CATALOG, and press RETURN. You will see a list of program names commonly called files that are on this "utility" disk. You can stop the scrolling of the listing by

pressing the CTRL key and W key simultaneously. Pressing these keys again will continue the listing. Type CATALOG again and use CTRL-W to stop the listing when the name ARCS.LOGO appears on the screen. ARCS is a Logo file that you will use in this activity. You must retrieve this file from the disk and place it in the computer's memory. To accomplish this task, press any key to complete the listing before typing the following.

```
READ "ARCS
```

You should receive the following message:

```
CIRCLEL DEFINED
ARCL DEFINED
ARCL1 DEFINED
CORRECTARCL DEFINED
CIRCLER DEFINED
ARCR DEFINED
ARCR1 DEFINED
CORRECTARCR DEFINED
?█
```

Enter the following commands and record your observations on the lines provided.

```
DRAW
REPEAT 360 [FD 1 RT 1] ____________________
PC -1 ______________________________________
REPEAT 360 [FD 1 RT 360/360] ______________
PC 1
REPEAT 360 [FD 1 LT 1] ____________________
```

2. Enter the following commands and describe the screen display.

```
DRAW
CIRCLE ____________________________________
CIRCLE R __________________________________
CIRCLER ___________________________________
CIRCLER 20 ________________________________
RT 90 _____________________________________
FD 20 _____________________________________
HT
```

NOTE The input value for CIRCLER is the radius of the circle drawn.

```
FD 20 ____________________________________
ST
LT 90 ____________________________________
PC -1
CIRCLEL 20 _______________________________
```

NOTE The line left on the screen is the diameter of the CIRCLEL 20.

```
LT 90 ____________________________________
FD 40 ____________________________________
SETH 0 ___________________________________
PC 1
```

3. Enter the following commands and observe the result.

```
DRAW
CIRCLEL 10
CIRCLEL 20
CIRCLEL 40
CIRCLEL 30
```

Describe the screen display. ____________________________________

4. Enter the following commands and record the results.

```
DRAW
LT 90 ____________________________________
PU _______________________________________
FD 10 ____________________________________
PD _______________________________________
RT 90 ____________________________________
CIRCLER 10 _______________________________
LT 90 ____________________________________
PU _______________________________________
FD 15 ____________________________________
PD _______________________________________
RT 90 ____________________________________
CIRCLER 25 _______________________________
LT 90 ____________________________________
```

```
PU ____________________________________________
FD 20 _________________________________________
PD ____________________________________________
RT 90 _________________________________________
CIRCLER 45 ____________________________________
LT 90 _________________________________________
PU ____________________________________________
FD 25 _________________________________________
PD ____________________________________________
RT 90 _________________________________________
CIRCLER 70 ____________________________________
LT 90 _________________________________________
PU ____________________________________________
FD 30 _________________________________________
PD ____________________________________________
RT 90 _________________________________________
CIRCLER 100 ___________________________________
PU HOME PD ____________________________________
f5 ____________________________________________
f3 ____________________________________________
```

5. Type the following and observe the screen display.

```
DRAW __________________________________________
CIRCLEL 40 ____________________________________
CIRCLER 40 ____________________________________
PU ____________________________________________
FD 80 _________________________________________
PD ____________________________________________
CIRCLER 40 ____________________________________
CIRCLEL 40 ____________________________________
PU HOME PD ____________________________________
HT ____________________________________________
ST ____________________________________________
```

6. Enter the commands and record the results.

```
DRAW
REPEAT 10 [CIRCLEL 30 RT 360/10] __________
```

```
DRAW ______________________________________
REPEAT 10 [CIRCLER 30 LT 36] ______________
f5 ________________________________________
f3 ________________________________________
DRAW
REPEAT 4 [CIRCLEL 75 LT 360/4] ____________
f5 ________________________________________
f3 ________________________________________
DRAW
REPEAT 7 [CIRCLER 40 RT 20] _______________
DRAW
REPEAT 50 [CIRCLEL 40 RT 360/50] __________
```

Turtle Talk. In graphics, a circle is represented as a regular polygon with many sides. The precision of the circle increases as the number of the sides of the polygon increases.

Part 1 of this activity demonstrates that with Commodore Logo, a circle can be represented as a regular polygon consisting of 360 sides in the procedure REPEAT 360 [FD 1 RT 1]. The circle is formed by repeating the steps, FD 1 RT 1, until the figure is closed. The forward movement of one turtle unit followed by a one degree rotation, 360 times, completes one revolution, thus satisfying the TOTAL TURTLE TRIP THEOREM.

The Logo Utility Disk includes two procedures for drawing circles in the ARCS file. The ARCS file is loaded after the system is booted with Logo as demonstrated in Part 1. CIRCLERight draws the circle clockwise. CIRCLELeft draws the circle counterclockwise. Both procedures require the input of the radius of the desired circle.

By entering CIRCLER 20 in Part 2, a circle with a radius of 20 units is drawn in a clockwise fashion. Once the circle is completed, the sequence [RT 90 FD 20] results in the turtle drawing a radius and stopping at the center of the circle. Another FD 20 command draws a diameter of the circle that passes through the unseen central point to an opposite point on the perimeter. This divides the circle in half.

Parts 4, 5, and 6 demonstrate a variety of circular designs which can be achieved by using the CIRCLER and CIRCLEL commands. Of particular interest is the use of the TOTAL TURTLE TRIP THEOREM to obtain interesting circular effects by consecutively rotating the turtle prior to drawing a circle. In order to complete the circular patterns, the number of degrees in each rotation is 360/N, where N is the number of times the drawing of the circle is repeated.

Turtle Truth. Study the following commands and predict the results. Enter the commands to confirm your prediction.

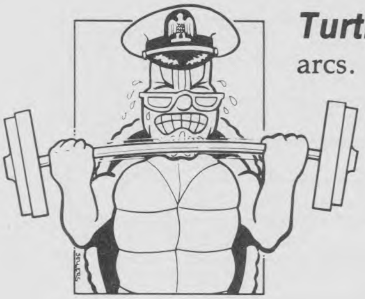
```
DRAW ________________________________
RT 90 _______________________________
FD 70 _______________________________
CIRCLER 30 __________________________
RT 90 _______________________________
FD 30 _______________________________
REPEAT 12 [FD 30 BK 30 RT 360/12] ___
LT 90 _______________________________
BK 30 _______________________________
LT 90 _______________________________
PU __________________________________
FD 30 _______________________________
PD __________________________________
FD 15 _______________________________
RT 90 _______________________________
FD 7 ________________________________
BK 14 _______________________________
FD 7 ________________________________
RT 90 _______________________________
FD 15 _______________________________
RT 90 _______________________________
FD 90 _______________________________
SETH 90 _____________________________
CIRCLER 30 __________________________
RT 90 _______________________________
FD 30 _______________________________
```

```
REPEAT 12 [FD 30 BK 30 RT 360/12] ________
LT 180 ______________________________
FD 30 _______________________________
RT 30 _______________________________
FD 50 _______________________________
SETH 135 ____________________________
FD 20 _______________________________
SETH 90 _____________________________
FD 10 _______________________________
BK 10 _______________________________
LT 135 ______________________________
FD 20 _______________________________
RT 90 _______________________________
FD 20 _______________________________
SETH 90 _____________________________
FD 10 _______________________________
PU HOME PD __________________________
HT __________________________________
```

Turtle Tester. Add more detail to the design of the drawing obtained in the Turtle Truth.

Turtle Teaser. Design the exterior of an expensive van or RV.

Activity 1.6 Turtle Arcs



Turtle Trials. This activity demonstrates how to draw arcs.

NOTE An arc is an incomplete or partially constructed circle. The length of an arc is measured in degrees as a fractional part of the circumference of the circle with a given radius. For example, an arc measure of 180 degrees is 180/360 or 1/2 of the circumference of a circle.

1. Use the procedure used in Part 1 of Activity 1.5 to enter the ARCS file from the Logo Utility Disk into the computer's memory. Enter the commands and record your observations on the lines below.

```
DRAW
CIRCLER 50 ____________________________
RT 90 _________________________________
FD 50 _________________________________
FD 50 _________________________________
LT 90 _________________________________
PC -1 _________________________________
ARC ___________________________________
ARCL __________________________________
ARCL 50 _______________________________
ARCL 50 180 ___________________________
ARCL 50 90 ____________________________
ARCL 50 10 ____________________________
ARCL 50 80 ____________________________
PC 1
```

NOTE The input 50 refers to the radius of the circle from which the arc is taken. The second input value refers to the measure of the arc that is desired.

2. Enter these commands and describe the results on the lines below.

```
DRAW
ARCR 75 120 ________________________________
RT 60 ______________________________________
ARCR 75 120 ________________________________
RT 60 ______________________________________
```

What is the sum of the number of degrees in each arc and the number of degrees in each RT turn? ______________________________

```
PR 120 + 120 + 60 + 60 __________________
REPEAT 2 [ARCR 75 120 RT (180 - 120)] ___
PR 120 + (180 - 120) ____________________
```

NOTE Parentheses are used to group arithmetic operations. Square brackets are used to indicate the start and end of Logo commands or lists. Be careful not to substitute ( ) with [ ].

3. Enter the following commands and record your observations.

```
DRAW
REPEAT 10 [ARCL 16 180 LT 180] __________
RT 180 __________________________________
```

4. Enter the following commands and observe the screen display.

```
DRAW
REPEAT 4 [ARCR 30 180 LT 90] ____________
ARCR 30 90 ______________________________
CIRCLER 60 ______________________________
f5 ______________________________________
f3 ______________________________________
```

5. Type the following commands and record the results.

```
DRAW
REPEAT 9 [ARCL 15 90 ARCR 15 90 ARCL 15 90
   ARCR 15 90 RT 160] ___________________
HT
```

Turtle Talk. The Commodore diskette contains two procedures for drawing circular arcs in the ARCS File. The ARCRight (ARCR) and ARCLeft (ARCL) commands require two input values. The first value defines the radius of the arc and the second value determines the length of the arc in arc degrees. Part 1 demonstrates that successive ARCL commands with a common radius (50), but with arc degree measures which total 360 degrees (180 + 90 + 10 + 80), eventually draw a circle. Note the sequence of commands is equivalent to the CIRCLEL 50 command.

The ARCR and ARCL are quite useful for drawing petal-like diagrams. The important principle to remember when drawing these shapes is the TOTAL TURTLE TRIP THEOREM. The total sum of arc degrees necessary to complete the entire petal is 360 degrees. The total sum of arc degrees necessary to complete one side of the petal is 180 degrees.

Figure 1-10 illustrates the relationship of critical turtle headings as the arc in Part 2 is drawn. At position A, the turtle has a heading of zero. Position B indicates that the heading is 120 degrees following the ARCR 75 120 command. Before drawing the bottom section of the petal (arc), the turtle heading must become 180, which is accomplished by the RT 60 command. REPEAT 2 [ARCR 75 120 RT 60] completes the petal and places the turtle in its original heading of zero.

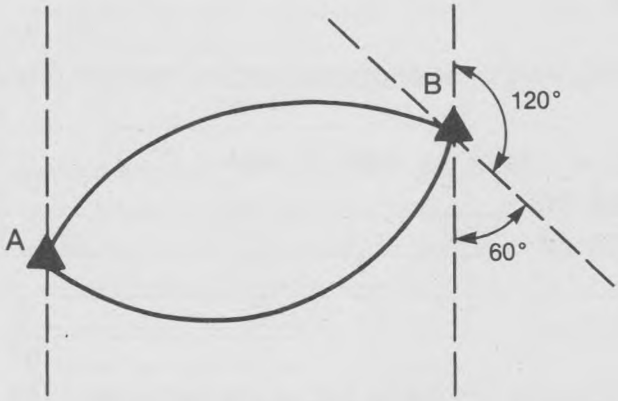


Figure 1-10.

Turtle Truth. Study the following commands and predict what the turtle is to draw. Check your prediction by entering the commands as given.

```
DRAW
PU
FD 50
RT 90
PD
CIRCLER 50
ARCR 50 45
SETH 0
REPEAT 2 [ARCR 70 100 RT 80]
PU HOME
FD 50
LT 90
PD
ARCL 50 45
SETH 360
REPEAT 2 [ARCL 70 100 LT 80]
PU HOME PD
```

Turtle Tester. Observe the outline of the animal's face on the screen. Add eyes, eyebrows, a mouth, and whiskers, then return the turtle to HOME to be the nose. Use ARCS and CIRCLES for as many facial features as possible.

Turtle Teaser. Draw an entire animal using several ARCS and CIRCLES.

Reference Point 1

Command Review. The following chart contains all of the abbreviations and definitions of the commands introduced thus far.

ABBREVIATIONS	DEFINITIONS
ST	Show Turtle.
HT	Hide Turtle.
FD N	Forward requires an input number N. For example, FD 10.
BK N	Back requires an input number N. For example, BK 20.
DRAW	Clears screen and returns turtle to HOME.
f5	Displays full graphics screen.
f1	Displays full text screen.
f3	Displays split screen with five lines of text and the remainder of the screen dedicated to graphics (normal mode).
RT N	Pivots turtle to its right and requires an input number N.
LT N	Pivots turtle to its left and requires an input number N.
SETH N	Points the turtle to the direction specified by the input number N. N is based on the clockwise revolution of the turtle in relation to HOME.

ABBREVIATIONS	DEFINITIONS
PR HEADING	Indicates the numerical value of the current SETH direction.
HOME	Positions turtle to the center of the screen at SETH 0.
PU	Permits movement without drawing.
PD	Places turtle in drawing mode.
PENERASE (PC −1)	Places turtle in erase mode equivalent to PENCOLOR -1.
PC 1	Sets color to white for drawing used to nullify the PENERASE mode.
CS	Erases drawing but does not affect position or heading of turtle.
REPEAT N [- - - - - -]	Runs a set of commands as many times as is indicated by the input number N.
[- - - - - -]	Brackets enclose a set of commands.
PR N	Prints the numerical value of N.
PR [- - - - - -]	Prints what is enclosed by the square brackets [ ].
CIRCLER R	A procedure in the ARCS file to draw a circle to the turtle's right and requires input of radius R.
CIRCLEL R	A procedure in the ARCS file to draw a circle to the turtle's left and requires input of radius R.
ARCR R S	A procedure in the ARCS file that draws segment of CIRCLER and requires input of radius R and input of arc size S.
ARCL R S	A procedure in the ARCS file that draws a segment of CIRCLEL and requires input of radius R and input of arc size S.

Application. Enter the following commands and study the resulting picture. Notice the programming techniques employed, especially the use of nested brackets (brackets within brackets) in the second line. With this structure, the [FD 30 RT 90] sequence is performed eight (4 * 2) times since the REPEAT 4 command happens twice. In contrast, the second FD 30 command is performed only twice; it is part of the REPEAT 2 sequence, but not part of the REPEAT 4 sequence.

```
DRAW
REPEAT 2 [REPEAT 4 [FD 30 RT 90] FD 30]
REPEAT 4 [FD 30 RT 90]
BK 30
LT 90
REPEAT 4 [FD 30 RT 90]
BK 60
REPEAT 4 [FD 30 RT 90]
RT 90
REPEAT 4 [FD 30 LT 45 FD 42 LT 45]
LT 135
FD 11
LT 90
REPEAT 4 [ARCR 10 180 LT 90 FD 11 RT 45 FD 30 RT
  45 FD 11 LT 90]
```

NOTE The computer continues a series of commands to the next line when the number of characters entered exceeds the screen limits. If the break occurs after a character, which is normally followed by a space, type the space on the new line. Failure to do so results in an error message.

```
RT 90
FD 31
RT 45
FD 11
REPEAT 3 [FD 10 LT 120]
BK 11
RT 90
FD 90
LT 90
REPEAT 3 [FD 30 RT 120]
FD 15
SETH 0
PU
FD 10
PD
CIRCLER 2.5
CIRCLEL 2.5
HT
```

Suggested Projects

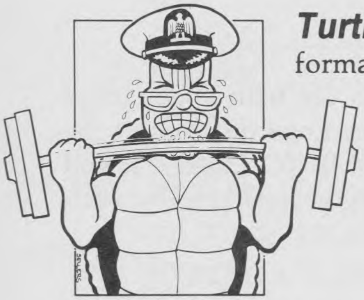
Try your hand at producing some of these objects. Keep a record of the commands you use. In Chapter 2 you will learn how to make these commands into a procedure and save them on a file disk. You will then be able to enter the procedure name and the computer will draw the entire picture for you.

AN IGLOO
GREEK COLUMNS
A VEHICLE
ICE CREAM SUNDAE
TELEVISION
COMPASS
SPECTACLES
A CACTUS
NUMERALS
CENTIPEDE
SAILBOAT
DIGITAL CLOCK
ANYTHING ELSE THAT STRIKES YOUR FANCY

Preserving Turtle Tracks

2

Activity 2.1 Turtle Procedure



Turtle Trials. This activity teaches the user how to format a file disk and define procedures.

1. In order to store information on a blank disk, you must first prepare the disk or "format" the disk by performing the following steps. It is only necessary to format the file disk once; from that point on you may use the file disk without repeating this entire procedure. Cold boot the computer with Logo the same way you have for all previous activities. When you receive the message

```
WELCOME TO LOGO!
?■
```

you will be ready to format a file disk on which to store information.

2. Remove the Logo disk from the disk drive and insert the blank disk you wish to format. Formatting this disk will erase any information that was previously stored on the disk. Be sure that you have inserted either a blank disk or one that you intend to erase and reuse. Type the following command:

```
DOS [N0:LOGO FILE DISK,01]
```

The red light on the disk drive will go on and you will hear several strange noises. Do not be alarmed. The disk is being formatted.

NOTE You may use any name in place of LOGO FILE DISK to identify the disk, but it must not be longer than 16 characters. You may also use any two digit number in place of 01. It is useful to title the disk with words that will be self explanatory for future reference. For example,

```
DOS [N0:LOGO CHAPTER 1,01]
```

3. When the cursor returns, type:

   ```
   CATALOG
   ```

   You will see the message

   ```
   LOGO FILE DISK

   664 BLOCKS FREE.
   ?■
   ```

   You now have a properly formatted file disk to use with Logo. In all future activities, please boot the system with the Logo disk, insert the file disk when you receive the message WELCOME TO LOGO! It will be assumed that you have completed the booting process prior to each activity.

NOTE There may be an occasion when you are using Logo that a "bug" or quirk in the program or system takes you out of Logo. Do not panic! There is a process that should restore Logo without rebooting. Simply hold down the RUN/STOP key, then press the RESTORE key. This should cause the message

```
WELCOME TO LOGO!
?■
```

to reappear on the screen. The work that you have not stored will be lost, but previously saved files will be intact.

4. Until now, you have used built-in commands such as FD, RT, and REPEAT. These commands are commonly called "primitives." You will now learn how to develop "procedures" by using the TO command. The TO command changes the screen display and puts Logo in "edit mode," which lets you teach the computer a procedure and give the procedure a name. Notice the highlighted edit line on the bottom of the screen. It is there to constantly provide you with the available options. Procedures that you provide in this mode can then be used like primitives. Enter the following commands and observe the screen display. Record your observations on the lines provided.

NOTE Commodore uses the CTRL key to perform special functions. The instruction CTRL-C means to press the CTRL and C keys simultaneously. Similarly, CTRL-G means to press the CTRL and G keys simultaneously. CTRL-G can be used whenever you "Goof" and desire to start over.

```
DRAW
REPEAT 4 [FD 70 RT 90] ____________________
DRAW
SQUARE ____________________________________
TO SQUARE _________________________________
REPEAT 4 [FD 70 RT 90] ____________________
END _______________________________________
CTRL-C ____________________________________
SQUARE ____________________________________
DRAW

REPEAT 4 [SQUARE RT 360/4] ________________
TO FOUR.SQUARE ____________________________
REPEAT 4 [SQUARE RT90]
CTRL-G
```

NOTE The command RT90 was an intentional mistake. By entering CTRL-G, the procedure we were about to define was interrupted or aborted, allowing for a new start. More sophisticated methods of correcting mistakes will be introduced later in this chapter.

```
TO FOUR.SQUARE ____________________________
REPEAT 4 [SQUARE RT 90]
END
CTRL-C
FOUR.SQUARE _______________________________
DRAW
REPEAT 8 [SQUARE RT 360/8] ________________
DRAW
TO EIGHT.SQUARE ___________________________
FULLSCREEN
REPEAT 8 [SQUARE RT 45]
END
```

```
CTRL-C ___________________________________
EIGHT.SQUARE _____________________________
f3 _______________________________________
```

NOTE As Logo does not allow spaces in names of procedures, we use a period (.) as an acceptable substitute to separate two-word names.

5. Type the following commands. Notice what happens and record the results.

```
DRAW
TO TRIANGLE ______________________________
REPEAT 3 [FD 80 LT 120]
END
CTRL-C ___________________________________
TRIANGLE _________________________________
DRAW
REPEAT 7 [TRIANGLE LT 360/7] _____________
TO SEVEN.TRI _____________________________
REPEAT 7 [TRIANGLE LT 360/7]
END
CTRL-C ___________________________________
DRAW
SEVEN.TRI ________________________________

DRAW
REPEAT 16 [TRIANGLE RT 360/16] ___________
TO SIXTEEN.TRI ___________________________
REPEAT 16 [TRIANGLE RT 360/16] ___________
END
CTRL-C ___________________________________
DRAW
SIXTEEN.TRI ______________________________
```

Turtle Talk. Prior to Activity 2.1, you were using only the Logo disk. You now have learned to prepare a file disk on which material developed in subsequent activities may be saved for future use. This second disk has been "formatted" for Logo. Parts 1, 2, and 3 guide you through the formatting process. These steps must be followed anytime you wish to make a new disk on which to store information.

In Chapter 1, all commands that are entered draw the diagram once and must be retyped each time you want the diagram to be drawn. At this point, you are working in what is referred to as "toplevel" mode. REPEAT 4 [FD 70 RT 90] in Part 4 serves as a review example, where this command draws a square only once. The command must be reentered each time a square is to be drawn.

In contrast, the TO command, introduced in Part 4, puts the user in "edit mode" where procedures can be developed. Creating a procedure means that you give a collection of commands a name. Now, by simply typing this name, the commands are performed. Part 5 defines the various procedures which are introduced in the activity. Four of these procedures, FOUR.SQUARE (Figure 2-1), EIGHT.SQUARE (Figure 2-2), SEVEN.TRI (Figure 2-3) and SIXTEEN.TRI (Figure 2-4) illustrate the principle of the TOTAL TURTLE TRIP THEOREM by drawing rotations of regular polygons defined by SQUARE and TRIANGLE. For example, SIXTEEN.TRI rotates TRIANGLE 360/16 degrees 16 times, which completes the total 360 degree drawing.

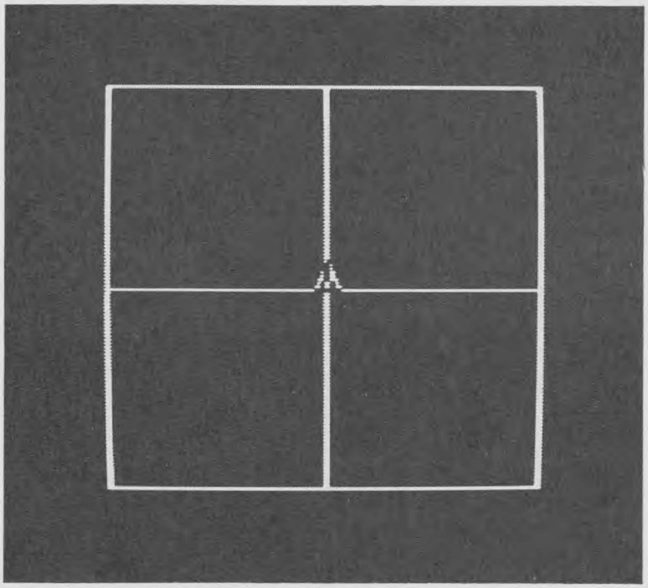


Figure 2-1.

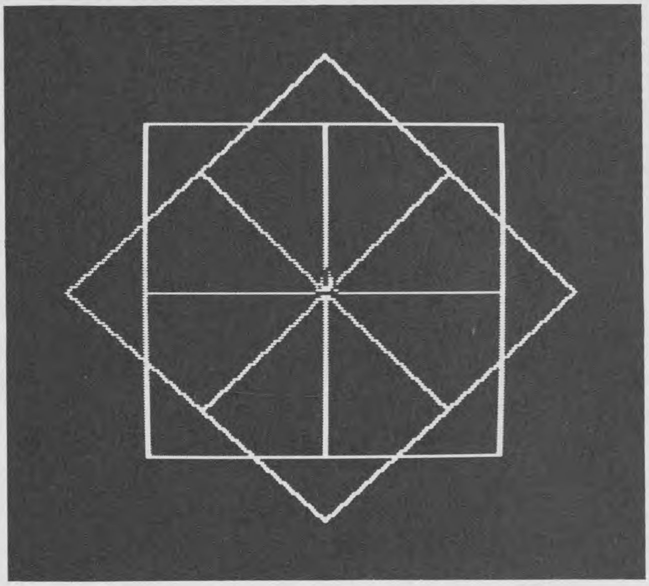


Figure 2-2.

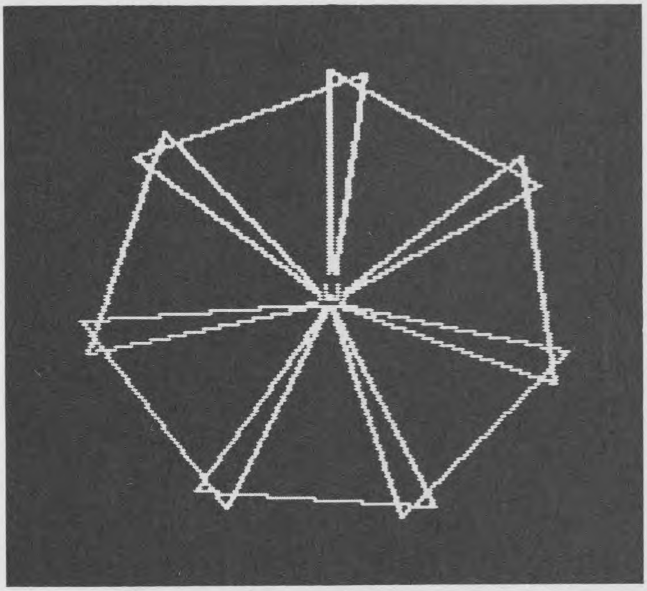


Figure 2-3.

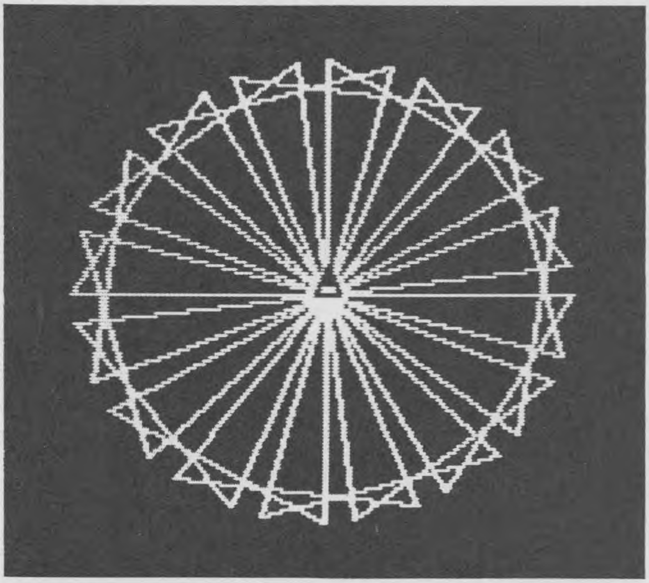


Figure 2-4.

In the procedure EIGHT.SQUARE, the command FULLSCREEN is entered. When the procedure is executed, this command will perform the same function as pressing the f5 key in top level mode. The function keys themselves cannot be used in edit mode, but the word that describes the effect of the function key may be entered as a command. Therefore, when working in edit mode, TEXTSCREEN and SPLITSCREEN must be used instead of f1 and f3, respectively. Once a procedure has changed the screen display option, it remains that way until you press a function key or execute a procedure containing a different option.

Turtle Truth. Study the following commands and predict the results. Type the commands, verify your predictions, and compare the screen display to the picture illustrated in Figure 2-5.

```
DRAW
FD 30
RT 90
FD 75
```

```
RT 90
FD 30
RT 90
FD 75
RT 90

TO RECTANGLE
REPEAT 2[FD 30 RT 90 FD 75 RT 90]
END
CTRL-C
DRAW
RECTANGLE
REPEAT 10 [RECTANGLE RT 360/10]
TO TEN.REC
FULLSCREEN
REPEAT 10 [RECTANGLE RT 36]
END
CTRL-C
DRAW
TEN.REC
```

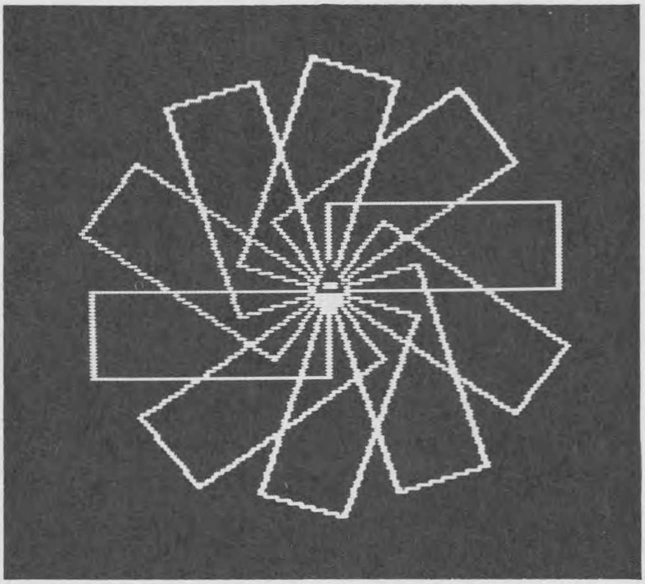


Figure 2-5.

Turtle Tester. The picture illustrated in Figure 2-6 is an example of a PENTAGON which has been rotated six times. Define a procedure called PENTAGON. Now define a procedure that rotates the PENTAGON a number of times to achieve a pleasing symmetrical design. (Answer on page 288.)

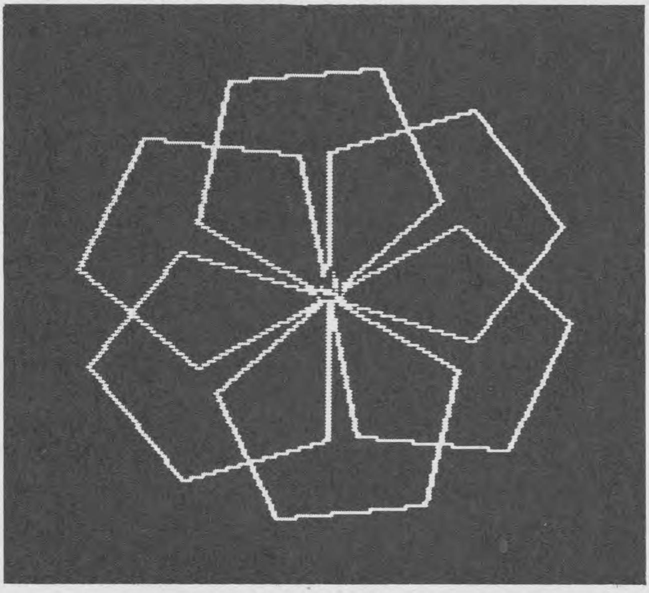


Figure 2-6.

Turtle Teaser. Define a procedure to draw the picture illustrated as Figure 2-7.

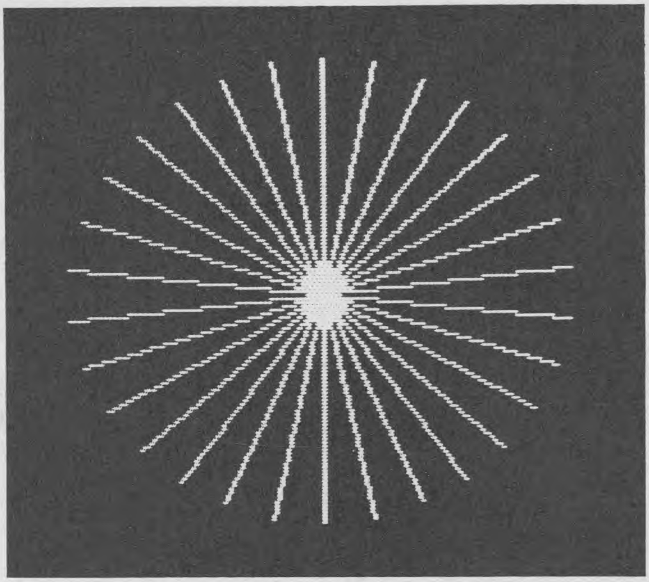
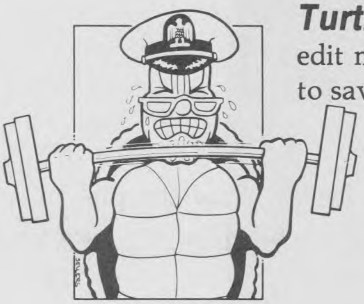


Figure 2-7.

Activity 2.2 Turtle Files



Turtle Trials. This activity reinforces the use of the edit mode to write procedures and then teaches you how to save them to a file disk for further use.

1. Cold boot the system with the Logo disk, insert your file disk when the screen display reads WELCOME TO LOGO!

2. Enter the following commands. Record the results after each command entered.

```
TO SQUARE ________________________________
REPEAT 4 [FD 30 RT 90]
END
CTRL-C ___________________________________
```

```
DRAW

SQUARE ________________________________________
FD 30 _________________________________________

SQUARE ________________________________________
FD 30 _________________________________________

SQUARE ________________________________________
FD 30 _________________________________________

f1 ____________________________________________
```

Review the commands just entered.

How many times are the commands SQUARE FD 30 entered? ________

```
f3 ____________________________________________

TO 3SQUARE ____________________________________
REPEAT 3 [SQUARE FD 30]
END
CTRL-C ________________________________________
3SQUARE _______________________________________
```

3. Enter the following commands and record the results.

```
BK 30 _________________________________________
RT 90 _________________________________________
BK 30 _________________________________________
3SQUARE _______________________________________
f1 ____________________________________________
```

NOTE This view of the previously typed text allows the user to observe the commands that were used to achieve the desired screen display. Because edit mode eliminates this screen display, it may be necessary to copy down the commands with pencil and paper. This helps you remember the sequence so that the procedure can be properly entered.

```
TO CROSS.SQUARE _____________________________
3SQUARE
BK 30
```

```
RT 90
BK 30
3SQUARE
END
CTRL-C ______________________________
CROSS.SQUARE ________________________
```

4. Enter the following commands and describe what happens.

```
RT 90 ______________________________
FD 30 ______________________________
RT 45 ______________________________
FD 30 ______________________________
HT ______________________________
FD 10 ______________________________
ST ______________________________
RT 45 ______________________________
FD 30 ______________________________
RT 45 ______________________________
FD 40 ______________________________
RT 45 ______________________________
FD 30 ______________________________
DRAW
```

5. Correct your previous estimations by typing the following commands. Record your observations.

```
CROSS.SQUARE ________________________
RT 90 ______________________________
FD 30 ______________________________
RT 45 ______________________________
FD 40 ______________________________
FD 2 ______________________________
HT ______________________________
ST ______________________________
RT 45 ______________________________
FD 30 ______________________________
RT 45 ______________________________
FD 42 ______________________________
```

```
RT 45 ________________________________________
FD 30 ________________________________________
f1 ___________________________________________
TO PERIMETER _________________________________
RT 90
REPEAT 4 [FD 30 RT 45 FD 42 RT 45]
END
CTRL-C _______________________________________
```

6. Enter the following commands and describe the results.

```
DRAW
CROSS.SQUARE _________________________________
PERIMETER ____________________________________
TO SHELL _____________________________________
CROSS.SQUARE
PERIMETER
END
CTRL-C _______________________________________
DRAW _________________________________________
SHELL ________________________________________
```

7. In order to reuse these procedures after shutting down the computer, we must save them on our file disk. Insert your file disk in the disk drive. Close the drive door, enter the following commands, and observe the results.

```
SAVETURTLE.PART ______________________________
SAVE TURTLE.PART _____________________________
SAVE "TURTLE.PART ____________________________
```

NOTE When you save a file, the red light on the disk drive glows while the file is being copied from the computer's memory to the disk.

```
SHELL ________________________________________
DRAW _________________________________________
GOODBYE ______________________________________
SHELL ________________________________________
CATALOG ______________________________________
```

Turtle Talk. In Activity 2.1, the concept of a procedure was introduced. This activity elaborates on the procedural approach to programming through the development of an example.

Logo is considered to be a "procedural" programming language. Procedures are small programs that collectively build a larger program. They allow for the breaking up of a complex figure into a number of small manageable parts. Using Logo terminology, procedures define new words or commands in terms of the primitives provided by the language itself. Procedures provide the possibility for a systematic approach to computer problem-solving. They are designed to minimize effort when finding and correcting logical errors.

In this activity, the complex figure (the shell of the turtle) is created by developing the components of the shell in sub-units that are used repetitively. The most primitive piece of the turtle shell is a square. Upon defining a procedure for drawing a SQUARE, three vertically adjacent squares are displayed and subsequently defined as a 3SQUARE procedure in Part 3. In turn, Part 4 demonstrates that 3SQUARE is used in the CROSS.SQUARE procedure to achieve the pattern desired for the top of the turtle shell, thereby completing the sequence of building components for the desired diagram. In Part 6, PERIMETER defines a procedure to connect all exterior vertices of the cross, thus completing the shape of the shell, which is illustrated in Figure 2-8. Therefore, the simplified procedure for SHELL is defined in Part 7, and requires only the use of CROSS.SQUARE and PERIMETER to complete the drawing of the turtle shell.

These newly created procedures are saved on disk for later access by the use of the SAVE command. TURTLE.PART is the arbitrary name given to the file that includes the five procedures you defined: SQUARE, 3SQUARE, CROSS.SQUARE, PERIMETER, and SHELL. As with procedure names, Logo only recognizes one word file names. A period can be used in place of a blank space for multi-word names. The name you choose for the file must be no longer than eleven characters. If it is, Logo will automatically shorten it to eleven.

The GOODBYE command erases all procedures currently in the computer's memory, in what is called the "workspace." Therefore, after GOODBYE has been used, any additions to this file or use of these procedures requires the user to retrieve this file from the disk and enter it into the workspace. Typing CATALOG verifies the existence of the file. This is an important topic in the next activity.

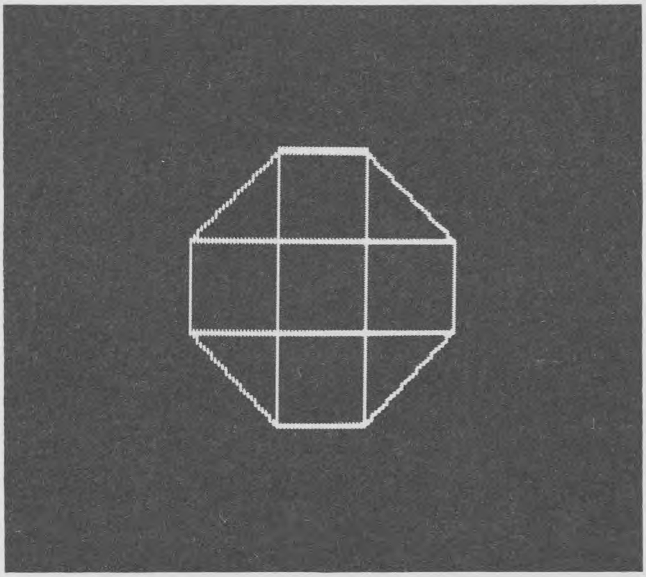


Figure 2-8.

Turtle Truth. Use the procedure employed in Part 1 of Activity 1.5 to enter the ARCS file from the Logo Utilities Disk into the computer's memory. Remove the Utilities Disk and insert your file disk. Study the following commands and predict the result. Enter the commands to confirm your prediction and compare the screen display with the picture illustrated in Figure 2-9.

```
TO HEAD
PU
FD 50
RT 90
PD
CIRCLER 50
END
CTRL-C
HEAD

ARCR 50 45
SETH 0
REPEAT 2 [ARCR 70 100 RT 80]
```

```
TO RIGHT.EAR
ARCR 50 45
SETH 0
REPEAT 2 [ARCR 70 100 RT 80]
END
CTRL-C

HEAD
RIGHT.EAR

PU
HOME
FD 50
LT 90
ARCL 50 45
SETH 360
PD
REPEAT 2 [ARCL 70 100 LT 80]

TO LEFT.EAR
PU
HOME
FD 50
LT 90
ARCL 50 45
SETH 360
PD
REPEAT 2 [ARCL 70 100 LT 80]
END
CTRL-C

HEAD
RIGHT.EAR
LEFT.EAR
PU HOME PD

TO WABBIT
HEAD
RIGHT.EAR
LEFT.EAR
PU HOME PD
END
CTRL-C
```

```
WABBIT

SAVE "WABBIT.PART
DRAW
WABBIT
GOODBYE
WABBIT
CATALOG
```

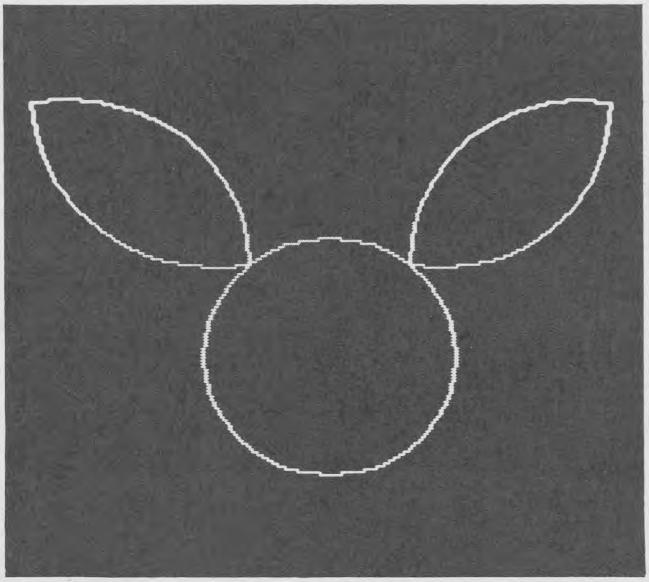
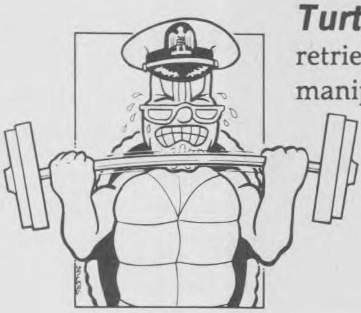


Figure 2-9

Turtle Tester. Refer back to Activity 1.4 and then define several procedures to draw a triangle, pentagon, hexagon, heptagon, octagon, nonagon, and decagon. Save these procedures in a file called POLYGONS. When you are done, erase the procedures from the workspace. Catalog the file disk to verify the existence of this new file. (Answer on page 289.)

Turtle Teaser. Clear the screen. Write procedures to draw uppercase letters. Save them under the file name CAPS. When you are done, erase the procedures from the workspace. The ambitious reader should be warned that this project could quickly exhaust the available memory in the workspace if the entire alphabet were to be created.

Activity 2.3 Turtle Inventory



Turtle Trials. This activity teaches the user how to retrieve a file from a previously created file disk and manipulate the procedures in that file.

1. Cold boot the computer with the Logo disk and file disk as usual. Type

```
CATALOG
```

NOTE If you saved all the procedures in Activity 2.2 under the suggested file names, you will see the following:

```
LOGO FILE DISK

#       TURTLE.PART.LOGO PRG
#       WABBIT.PART.LOGO PRG
#       POLYGONS.LOGO    PRG
#       CAPS.LOGO        PRG
#       BLOCKS FREE.
```

The # sign preceding each file stands for a number that indicates the amount of space occupied by each file on the disk. The magnitude of the number may vary depending on how many procedures are in the file and the length of each procedure. The BLOCKS FREE message indicates the amount of unused space which remains on the disk.

2. Type the following commands. Wait for the cursor to reappear after each entry. Observe and record the results.

```
READ TURTLE.PART.LOGO ____________________
READ "TURTLE.PARTS.LOGO
GOODBYE __________________________________
CATALOG __________________________________
READ "TURTLE.PART ________________________
NODRAW ___________________________________
POTS _____________________________________
```

NOTE POTS stands for Print Out all TitleS.

```
SQUARE ___________________________________
DRAW
3SQUARE __________________________________
DRAW
CROSS SQUARE _____________________________
POTS _____________________________________
f1 _______________________________________
CROSS.SQUARE _____________________________
f3
f1
DRAW
f5
PERIMETER ________________________________
f1
DRAW
CROSS.SQUARE _____________________________
f3 _______________________________________
PERIMETER ________________________________
DRAW
SHELL ____________________________________
```

3. Enter the following commands and record the results.

```
GOODBYE __________________________________
```

```
POTS ______________________________________
CATALOG ___________________________________
READ "WABBIT.PART _________________________
WABBIT ____________________________________
READ "TURTLE.PART _________________________
SHELL _____________________________________
PU HOME PD ________________________________
WABBIT ____________________________________
ERASE ALL _________________________________
DRAW ______________________________________
```

NOTE ERASE ALL performs the same function as GOODBYE.

4. Enter the following commands and describe the screen display.

```
POTS ______________________________________
f1
DRAW ______________________________________
SHELL _____________________________________
READ "TURTLE.PART _________________________
PO ALL ____________________________________
```

NOTE PO ALL stands for Print Out ALL procedures.

```
CLEARTEXT _________________________________
POTS ______________________________________
PO "3SQUARE _______________________________
PO 3SQUARE ________________________________
3SQUARE ___________________________________
PO SHELL __________________________________
f1 ________________________________________
```

NOTE PO stands for Print Out.

```
GOODBYE ___________________________________
POTS ______________________________________
```

5. Enter the following commands and record the results.

```
CATALOG ___________________________________
```

```
READ "TURTLE.PART ______________________________
SAVE "TORTOISE _________________________________
GOODBYE ________________________________________
POTS ___________________________________________
CATALOG ________________________________________
READ "TORTOISE _________________________________
SHELL __________________________________________
ERASE ALL ______________________________________
POTS ___________________________________________
CATALOG ________________________________________
ERASEFILE "TORTOISE ____________________________
CATALOG ________________________________________
READ "TURTLE.PART ______________________________
```

Turtle Talk. Activities 2.1 and 2.2 demonstrate how to create and save procedural files on disk. In this activity your concern is how to retrieve this information in an organized way.

When you boot the computer with the Logo disk and eventually receive the "WELCOME TO LOGO" message, you have a workspace (or part of computer memory) to use. Initially this workspace is empty. As you define procedures and variables, this information will accrue in the workspace. This storage is temporary in the sense that when you turn the computer off, all information in the workspace is lost unless you first save the information on a disk. The SAVE command introduced in Activity 2.1 transfers all procedures accumulated in the workspace to disk. Files may be retrieved from a disk and put into the workspace by using the READ command. In Part 2, either READ "TURTLE.PART or READ "TURTLE.PART.LOGO takes all procedures contained in the file named TURTLE.PART from the disk and makes them available for use in the workspace.

A problem that frequently confronts the Logo user is a lapse of human memory. It is difficult to remember the names of all files, names of the procedures within the particular files, and finally, the list of commands that define the various procedures. Fortunately, there are several commands to assist the user with managing this information.

When the user cannot remember the names of files on a disk, he can use the CATALOG command introduced in Part 3. The CATALOG command provides a list of all names of Logo files that are contained on the disk. Notice that each file name ends with .LOGO. The suffix serves as an indicator that the file is unique to Logo. It is not necessary to type .LOGO when READing or SAVEing a file. Files that are of no further use can be removed from the disk by using the ERASEFILE command. ERASEFILE "TORTOISE in Part 5 demonstrates the use of this command.

Once a file has been READ into the workspace, a listing of the procedures in that file are immediately placed on the screen. POTS (Print Out TitleS) also provides a listing of all names or titles of procedures within the workspace at that time. The POTS command is particularly useful when you need to review the procedures that are contained within that file. The NODRAW command is used to erase text from the screen.

There is, however, one peculiarity of interest. Notice in Part 3 that the procedures from different files can accumulate in the workspace if successive READ commands are used. The READing of one file does not erase and replace the contents of the workspace. Instead, the new procedures are appended to the existing procedures. This occurrence may or may not be desirable. In the event you wish an empty workspace prior to READing a file, the GOODBYE or ERASE ALL commands are available. Part 3 demonstrates the use of both commands prior to each READ command to assure a clean workspace.

Finally PO ALL (Print Out ALL procedures) is a command that lists the names and definitions of all procedures contained in the workspace. Part 4 demonstrates that one limitation of the PO ALL command is that if the file is lengthy, the listing scrolls off the screen too quickly for viewing. However, the PO (Print Out) command permits the user to observe the listing of commands for a particular procedure, such as PO SHELL. Also, although not demonstrated in the activity, you can use CTRL-W to freeze the scrolling of text. Typing any key will cause the listing to continue. The CLEARTEXT command and f1 key conveniently clear the screen and devote it entirely to text for a less congested view of the listing.

Complete familiarity with these file and workspace management commands is essential to working efficiently with Logo. Regular use of these functions will help the user develop confidence and expertise.

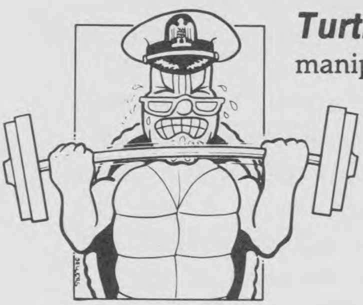
Turtle Truth. Study the following commands and predict the outcome. Enter the commands to confirm your prediction. Cold boot the system with Logo. Insert your file disk.

```
CATALOG
READ "WABBIT.PART
PO ALL
WABBIT
PO WABBIT
f1
f5
f3
SAVE "RABBIT.PART
GOODBYE
READ "RABBIT.PART
WABBIT
ERASEFILE "RABBIT.PART
CATALOG
READ "WABBIT.PART
f1
CLEARTEXT
POTS
ERASE ALL
POTS
F3
POTS
```

Turtle Tester. Start with the computer off. Cold boot the system with Logo and then insert your file disk. Call up the catalog of files. Read in the file POLYGONS (created in Activity 2.2). Examine a list of the procedures in that file. Run one of the procedures. Remove the current information in workspace. Call up the catalog again to verify the existence of the original file.

Turtle Teaser. Perform similar operations on the file CAPS that was created in Activity 2.2.

Activity 2.4 Turtle Editing



Turtle Trials. This activity demonstrates how cursor manipulation can be used to edit existing procedures.

1. Boot the system with Logo as usual. Remove the disk, insert the Utilities Disk, and read in the file ARCS. Enter the following procedure.

```
TO TRACK
HT
PU
FD 50
RT 90
BK 50
PD
REPEAT 2 [FD 100 ARCR 50 180]
END
CTRL-C

TRACK

EDIT TRACK
```

The Logo editor allows the manipulation of the cursor to effect changes in the procedure held in a reserved area of memory known as the "edit buffer."

NOTE Throughout the remainder of this activity, you will be learning to make use of CTRL keys for the purpose of editing previously entered Logo commands. The manner in which you enter CTRL commands is different from the manner in which you enter Logo commands. The RETURN key is pressed after entering each Logo command. The RETURN key is not pressed after entering a CTRL command.

Observe the contents of the "edit buffer" and record the results of the following CTRL commands. Do not press RETURN after each command or you will have unwanted movement of the text.

```
CTRL-N ______________________________
CTRL-N ______________________________
CTRL-N ______________________________
CTRL-P ______________________________
CTRL-P ______________________________
CTRL-O ______________________________
CTRL-N ______________________________
CTRL-N ______________________________
CTRL-O ______________________________
CTRL-P ______________________________
CTRL-P ______________________________
CTRL-P ______________________________
CTRL-G ______________________________
EDIT TRACK __________________________
CTRL-C ______________________________
TRACK
```

What is the function of CTRL-N? ______________________

What is the function of CTRL-P? ______________________

What is the function of CTRL-O? ______________________

What is the function of CTRL-G? ______________________

What is the function of CTRL-C? ______________________

2. There are two CRSR keys located at the lower right corner on the Commodore keyboard. The left-most key gives cursor movement either "up" or "down." The right-most key provides "left" or "right" cursor movement. A normal CRSR keypress gives "down (↓)" or "right (→)" cursor movement. Pressing the SHIFT key simultaneously with a CRSR key gives "up (↑)" or "left (←)" cursor movement.

   Using the keypresses defined above, describe the movement caused by the following instructions.

```
EDIT TRACK
→ ________________________________
→ ________________________________
→ ________________________________
```

→ ____
→ ____
`CTRL-A` ____________________________________
`CTRL-L` ____________________________________
← ____
← ____
← ____
`CTRL-F` ____________________________________
`CTRL-P` ____________________________________
`CTRL-P` ____________________________________
`CTRL-B` ____________________________________
`CTRL-C` ____________________________________

What is the function of → ? ______________________________
What is the function of ← ? ______________________________
What is the function of CTRL-A? ___________________________
What is the function of CTRL-L? ___________________________
What is the function of CTRL-B? ___________________________
What is the function of CTRL-F? ___________________________

3. Describe the movement produced by each of the following edit commands.

`EDIT TRACK`
`CTRL-N` ____________________________________
`CTRL-K` ____________________________________
`CTRL-K` ____________________________________
`CTRL-N` ____________________________________
`CTRL-N` ____________________________________
`CTRL-N` ____________________________________
`CTRL-N` ____________________________________
`CTRL-O` ____________________________________
`HT` ____
`CTRL-A` ____________________________________
`CTRL-D` ____________________________________
`CTRL-D` ____________________________________
`CTRL-D` ____________________________________
`CTRL-G` ____________________________________

What is the function of CTRL-D? ___________________________
What is the function of CTRL-K? ___________________________

Turtle Talk. To alleviate the frustration of modifying procedures, Commodore Logo provides several editing commands that let the user move the cursor on the screen to make additions and deletions. These editing commands are categorized and summarized below.

Horizontal Control

← moves the cursor one space to the left and does not erase a character each time the right CRSR key is pressed simultaneously with the SHIFT key.

→ moves the cursor one space to the right each time the right CRSR key is pressed without erasing any characters.

CTRL-A moves the cursor to the beginning of the line in which it is positioned.

CTRL-L moves the cursor to the end of the Line in which it is positioned.

Vertical Control

CTRL-N moves the cursor down to the Next line from its current position.

CTRL-P moves the cursor up to the Previous line from its current position.

CTRL-O lowers subsequent lines of the procedure to Open a new line at the current cursor position.

CTRL-F moves the cursor Forward one entire screen of text or to the end of the procedure, whichever comes first.

CTRL-B moves the cursor Back one entire screen or to the beginning of the edit buffer.

Text Manipulators

CTRL-D Deletes the character under the cursor.

CTRL-K deletes or Kills all characters on the line to the right of the cursor.

Workspace Management

ER - - - - - - is used to ERase the designated procedure from the workspace. For example,

```
ER SQUARE
```

Table 2-1 illustrates the various functions and cursor positions for the editing commands discussed in this activity.

Table 2-1. Editing commands with their various functions and cursor positions.

Edit Commands	Before Using	After Using
←	FORWARD 30■	FORWARD 30
→	FORWARD 30	FORWARD 30
CTRL-A	FORWARD 30■	FORWARD 30
CTRL-L	FORWARD 30	FORWARD 30■
CTRL-N	FORWARD 30 RT 90	FORWARD 30 RT 90
CTRL-P	FORWARD 30 RT 90	FORWARD 30 RT 90
CTRL-O	FORWARD 30 RT 90	FORWARD 30 ■ RT 90
CTRL-D	FORWARD 30	FORARD 30
CTRL-K	FORWARD 30	F■

Turtle Truth. Describe the function and movement of the cursor for each of the following commands when used collectively. Remember not to use the RETURN key after each command when working in the edit mode.

```
EDIT TRACK
→ (3 TIMES)
BIG
CTRL-N
CTRL-A
```

```
→
CTRL-D
S
CTRL-N (6 TIMES)
CTRL-L
← (7 TIMES)
CTRL-D
6
← (9 TIMES)
CTRL-D
2
```

ATTENTION Your procedure should now be identical to Checkpoint 2-1. If it is not, continue editing.

```
TO BIGTRACK
PU
ST
FD 50
RT 90
BK 50
PD
REPEAT 2 [FD 120 ARCR 60 180]
HT
END
```

Checkpoint 2-1.

```
CTRL-C
BIGTRACK
PU HOME PD
TRACK
f1
POTS
ER BIGTRACK
POTS
```

NOTE Since BIGTRACK was developed for editing practice only, it is unnecessary to retain it for further use. ER BIGTRACK removes BIGTRACK from the workspace.

Turtle Tester STADIUM uses TRACK and other newly defined procedures to draw a football field with a football in the end zone. Enter the procedures, then type STADIUM to observe the diagram illustrated in Figure 2-10. Perform the following editing exercises and observe the gross distortions that occur with slight modifications to the existing procedures.

```
TO STADIUM
TRACK
LINES
PU BK 80 LT 90 FD 10 PD
SEAT
PU BK 130 PD
SEAT
PU FD 50 LT 90 FD 50 RT 90 PD
CROSSBAR
PU BK 35 RT 90 FD 135 LT 90 PD
CROSSBAR
PU BK 15 RT 90 FD 15 LT 90 PD
FOOTBALL
END

STADIUM

TO LINES
REPEAT 11 [ RT 90 FD 100 BK 100 LT 90
 FD 10]
END

LINES
DRAW
STADIUM

TO SEAT
REPEAT 2 [FD 10 RT 90 FD 40 RT 90]
END

TO CROSSBAR
BOX
FD 5 RT 90 FD 3 LT 90
FD 30
LT 90 FD 3 RT 90
BOX
END

CROSSBAR

TO BOX
REPEAT 4 [FD 5 RT 90]
END

CROSSBAR
DRAW
STADIUM

TO FOOTBALL
REPEAT 2 [ARCR 6 120 RT 60]
END

FOOTBALL

DRAW
STADIUM
SAVE "GRIDIRON
```

Editing Exercises

1. Edit STADIUM, use CTRL-N and CTRL-K in conjunction with other commands to alter the procedure to read as follows:

```
TO STADIUM
TRACK
LINES
PU BK 80 LT 90 FD 10 PD
SEAT
PU BK 130 PD
SEAT
PU FD 50 LT 90 FD 50 RT 90 PD
CROSSBAR
FOOTBALL
END
```

2. After the procedure name LINES in STADIUM, use CTRL-K, CTRL-Y,

CTRL-P, and CTRL-O sequences so the procedure appears in the following form.

```
TO STADIUM
TRACK
LINES
FOOTBALL
PU BK 80 LT 90 FD 10 PD
SEAT
PU BK 130 PD
SEAT
PU FD 50 LT 90 FD 50 RT 90 PD
CROSSBAR
END
```

3. Edit the procedure LINES, using CTRL-N, CTRL-D and other editing commands so the number 11 is replaced with the number 10. It should be altered to appear as follows:

```
TO LINES
REPEAT 10 [RT 90 FD 100 BK 100 LT 90 FD 10]
END
```

4. In the procedure CROSSBAR, add ST immediately after the first line and insert HT just before the END command. Use CTRL-O in conjunction with appropriate editing commands so that CROSSBAR is in the following form.

```
TO CROSSBAR
ST
BOX
FD 5
RT 90
FD 30
LT 90
FD 3
RT 90
BOX
HT
END

STADIUM
```

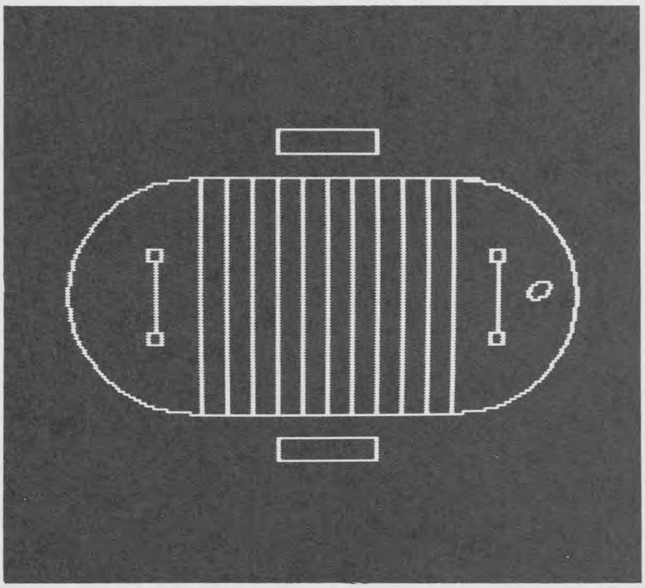


Figure 2-10.

Turtle Teaser. The following procedures attempt to draw the baseball diagram illustrated in Figure 2-11. Enter all of the following procedures. Type BASEBALL to observe the results. There are several errors to be edited. The arrows (→) will assist you in locating these mistakes. Use the edit mode to make appropriate corrections. Begin by inserting the Utilities Disk and reading in the ARCS file.

```
   TO BASEBALL
   HT
   BOUNDARY
   PU RT 135 FD 60 PD SETH 0
   INFIELD
→  MOUNT
   PU BK 48 LT 45 PD
   BASE SETH 0
   PU LT 45 FD 70 PD RT 90
→  BASE
→  PU FD 70 RT 90 PD
   BASE
   PU FD 70 RT 90 PD
   BASE
```

```
     PU FD 70 RT 90 PD
     BASE SETH 0
     PU LT 20 FD 15 PD SETH 0
     PLAYER
     PU LT 35 FD 35 PD SETH 0
     PLAYER
     PU RT 55 FD 50 PD SETH 0
     PLAYER
     PU LT 60 FD 75 PD SETH 0
     PLAYER
     PU LT 120 FD 70 PD SETH 0
     PLAYER
     PU RT 135 FD 45 PD SETH 0
→
     PU LT 140 FD 30 PD SETH 0
     PLAYER
     PU RT 103 FD 46 PD SETH 0
     PLAYER
→    PU RT 90 FD 65 PD SETH 0
     PLAYER
     END

     TO BOUNDARY
→
     BK 65
     LT 45
     PD
     FD 130
→
     RT 90
     FD 130
     LT 45
     ARCL 92 180
     SETH 0
     END

     TO INFIELD
     RT 45
     FD 70
     RT 90
     FD 70
     LT 135
     END
```

```
         TO MOUND
         PU LT 90 FD 49 PD
         CIRCLER 10
         SETH 0
         END

         TO BASE
    →    REPEAT 4 [FD 8 RT 85]
         END

         TO PLAYER
         CIRCLER 3
         END
         CTRL-C

         BASEBALL
```

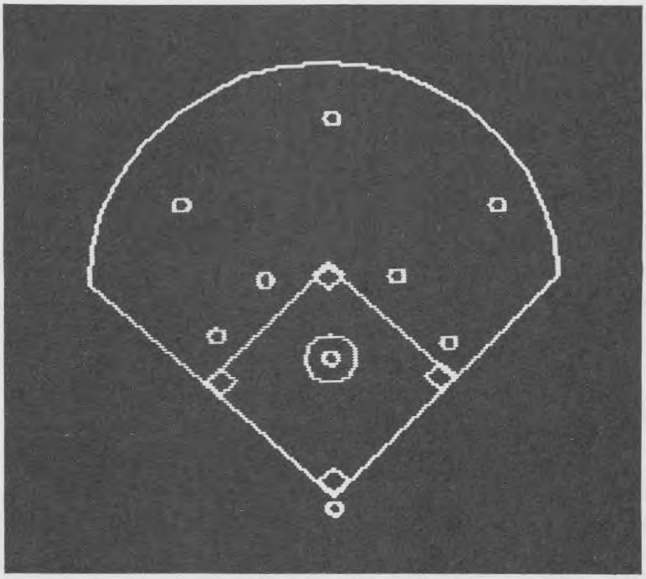
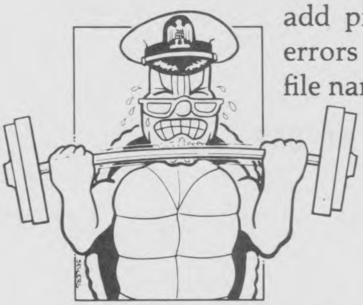


Figure 2-11

NOTE Once the editing is completed and the procedure BASEBALL runs without any bugs, insert your file disk and save this file as BALLPARK. Clear the workspace with GOODBYE.

Activity 2.5 Turtle Controls



Turtle Trials. This activity teaches the user how to add procedures to an existing file, how to correct any errors in procedure definitions, and how to use a utility file named TEACH to assist in defining procedures.

1. Cold boot the computer with the Logo disk as usual. Use the Utilities Disk and read in the file ARCS and the file TEACH. Then insert your file disk. Type the following commands and record your observations. The CATALOG command lets you see the listing of file names on your disk.

```
CATALOG ________________________________
READ "TURTLE.PART ______________________
SHELL __________________________________
```

NOTE Please assume that each procedure definition should be concluded by typing CTRL-C after END. From this point on, it will not be printed in the instructions unless several procedures are being defined together in the edit buffer.

2. Enter the following procedure.

```
TO TURTLE
SHELL
PAWS
TAIL
HEAD
PU HOME PD
END
```

3. Now type in the following commands to observe the turtle and its new parts.

```
TURTLE _________________________________
FD 30
RT 45
```

```
FD 11
LT 90
ARCR 10 180
LT 90
FD 11
RT 45
FD 30
RT 45
FD 11
LT 90
ARCR 10 180
```

4. Enter the following commands and observe that the TEACH file lets you interactively define a procedure without leaving toplevel mode. The screen isn't cleared as it was when the editor was used, therefore you can review the previously typed text by pressing f1, or observe the picture and five lines of text by pressing f3. Begin by typing

   ```
   TEACH
   ```

   After pressing RETURN the computer responds with

   ```
   NAME:
   ```

   And waits for you to type the name of the procedure. Type the name PAWS.

   ```
   NAME:PAWS
   ```

   After pressing RETURN, the computer responds with the following message.

   ```
   TO PAWS
   >
   ```

   The > prompt indicates where you are to enter each command in TEACH mode. Now complete procedure PAWS by entering the following commands.

NOTE If you make any typing errors, you must complete the procedure, then correct the errors using normal editing methods.

```
f1 ________________________________________
FD 30
RT 45
FD 11
LT 90
```

```
ARCR 10 180
LT 90
FD 11
RT 45
FD 30
RT 45
FD 11
LT 90
ARCR 10 180
END ______________________________
f3 _______________________________
DRAW _____________________________
TURTLE ___________________________
```

NOTE You are now going to be entering both Logo procedure commands and CTRL key editing commands. The use of the RETURN key gets very tricky at this point. You must remember *not* to press the RETURN key after a CTRL key used for editing purposes. But you should press the RETURN key after entering a Logo procedure command. To assist your memory in this activity, we shall cue you with (n-r), which stands for "no return", following each editing command. This is to remind you not to press RETURN after using that CTRL key. If there is no (n-r), press RETURN as usual.

5. Type in the following commands and note the results.

```
EDIT PAWS ________________________
CTRL-N(5 TIMES) (n-r) ____________
REPEAT 4 [CTRL-L LT 90 FD 11 RT 45 FD 30
  RT 45 FD 11 LT 90]
CTRL-K (press this combination of keys
  until the cursor is on the E of the word
  END) (n-r)
CTRL-C (n-r) _____________________
TURTLE ___________________________
```

6. Enter the following commands and record the results. Commands preceded by an * indicate intentional errors. Type these commands exactly as they appear. You will correct them later.

```
RT 90 ____________________________
```

```
  FD 31 ____________________________________
  RT 45 ____________________________________
  FD 11 ____________________________________
  REPEAT 3 [FD 10 LT 120] __________________
  HT _______________________________________
  ST _______________________________________
  TEACH ____________________________________
  NAME: TAIL _______________________________
  f1
  RT 90 ____________________________________
* FD _______________________________________
  CTRL-P ___________________________________
  CTRL-A ___________________________________
  CTRL-K ___________________________________
* FD 111 ___________________________________
  ←     ____________________________________
* REAPEAT 3 [FD 10 LT 120] _________________
  END ______________________________________
  f3 _______________________________________
  DRAW _____________________________________
  TURTLE ___________________________________
  EDIT TAIL ________________________________
  CTRL-N (n-r) _____________________________
  CTRL-N (n-r) _____________________________
  CTRL-L (n-r) _____________________________
  push the SPACEBAR once (n-r) _____________
  31 (n-r) _________________________________
  CTRL-A (n-r) _____________________________
  CTRL-N (n-r) _____________________________
  CTRL-O (n-r) _____________________________
  RT 45 (n-r) ______________________________
  CTRL-N (n-r) _____________________________
  CTRL-D (n-r) _____________________________
  CTRL-N (n-r) _____________________________
  ←    (n-r) _______________________________
  ←    (n-r) _______________________________
  CTRL-D (n-r) _____________________________
```

```
    CTRL-C (n-r) ____________________________
    TURTLE ____________________________________
```

7. Enter the following and record the results.

```
    POTS ____________________________________
    f1 ______________________________________
    SAVE "TURTLE.PART _______________________
    TURTLE __________________________________
    GOODBYE _________________________________
    TURTLE __________________________________
    CATALOG _________________________________
```

Turtle Talk. In Activity 2.4 you were introduced to and practiced with the Logo editor as a means by which commands can be added, deleted, or modified. In this activity, this newly acquired skill is used during the development of the turtle illustrated in Figure 2-12. The activity emphasizes what is commonly called the top-down procedure building approach by separating the turtle into several subsidiary components; namely, a shell, four paws, a tail, and a head.

In Part 2, a procedure called TURTLE defines the sequence of subprocedures by which the turtle is drawn. Part 3 demonstrates how experimenting with commands can serve as a prelude to the definition of a procedure, such as PAWS in Part 4.

The file named TEACH, that is read in from the Utilities Disk, is introduced to demonstrate the value of being able to review the commands entered during the experimental stage of designing a procedure. Being able to view these commands eliminates the need to copy them on paper before entering edit mode. TEACH allows you to define a procedure without clearing any screen. Typing f1, f3, or f5 shows you TEXTSCREEN, SPLITSCREEN, or FULLSCREEN, respectively. It is important to read TEACH from the Utilities Disk prior to typing any instructions that you might want to copy because loading the file clears the screen.

Part 5 uses the editor to simplify the length of PAWS. Parts 6 and 7 develop the remaining procedures for TURTLE, with the assistance of the editor to correct several deliberate errors in the procedure listings. When using TEACH to define procedure TAIL in Part 6, editing commands used to move from line to line do not work, but those used within a line do work. To correct line to line mistakes, one must put the file in the edit buffer.

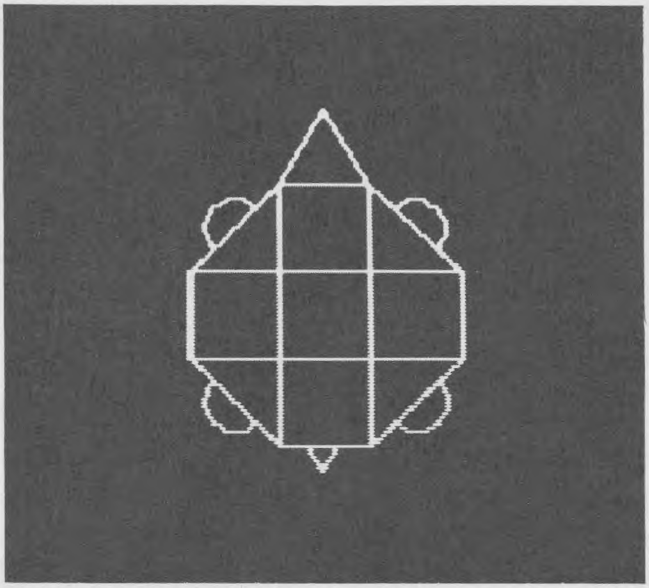


Figure 2-12.

Turtle Truth. Study the following commands and predict the result. Type them in to verify your prediction.

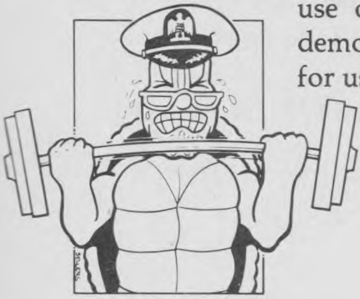
```
READ "TURTLE.PART    ________________________
TURTLE    ___________________________________
BK 11    ____________________________________
RT 90    ____________________________________
FD 90    ____________________________________
LT 90    ____________________________________
REPEAT 3 [FD 30 RT 120]    __________________
TEACH    ____________________________________
NAME:HEAD    ________________________________
f1    _______________________________________
BK 21 (n-r)
←  (n-r)    _________________________________
←  (n-r) ______________________________________
CTRL-D (n-r)
1
RT 90
FD 90
LT 90
REPEAT 3 [FD 30 RT 120]
END

f3 ____________________________________________
DRAW __________________________________________
TURTLE ________________________________________
EDIT HEAD
CTRL-N (6 times)(n-r)
CTRL-O (n-r)
HT
CTRL-C
TURTLE ________________________________________
SAVE "TURTLE.PART
GOODBYE
DRAW
TURTLE
CATALOG
f1
```

Turtle Tester. Read in the file TURTLE.PART. Have it draw the TURTLE. Show the turtle. Experiment and draw eyes on the turtle. Use TEACH to define a procedure called EYES but do not HT at the end. Put the procedure EYES into edit mode and move the cursor so that you can add HT on the line before END. Have it draw the TURTLE. Replace the old file called TURTLE.PART by saving the new TURTLE.PART that includes the new EYES procedure. Have it draw the turtle. Erase the file from the workspace or computer memory.

Turtle Teaser. Add the procedures to make a face on the WABBIT in the file called WABBIT.PART. Save the new file on the disk. Clear it from your workspace.

Activity 2.6 Turtle Maintenance

Turtle Trials. This activity reinforces the appropriate use of previously introduced CTRL characters. It also demonstrates that selected CTRL characters are available for use outside of the edit mode.



1. Cold boot the system as usual. Read in the TEACH file from the Utilities Disk, then insert your file disk. Type the following commands and observe the results.

   `CATALOG` ____

   `POTS` ____

   `DRAW` ____

NOTE Typing CATALOG lets you see a listing of the files on this disk. By not loading any of these files, you are choosing to create new material. The POTS command indicates that the only procedures in workspace are the three read in from the TEACH file.

2. You are now going to experiment with some commands and CTRL characters in toplevel mode. In toplevel you are directing the turtle's immediate movements, rather than creating a predefined procedure to control its path.

As you work through the following parts of this activity, you will be asked to make some intentional errors so they can be corrected by using the edit mode CTRL characters previously introduced. In Part 1 you will see that several of these CTRL characters work in toplevel, but some of them do not. The ones that do not work will result in no change of screen display.

As you discover which CTRL characters are available for use at toplevel, circle them. Notice some are only available under certain circumstances. Cross out the CTRL characters that do not work at toplevel.

NOTE Press RETURN after you have entered an entire line of commands unless instructed otherwise with the (n-r) indicator. The combination of CTRL keys presented on one line is to demonstrate how several keys can be used collectively to produce desired editing results. The INST/DEL instruction means to press the key located at the upper right corner of the keyboard labeled with INST and DEL.

```
HT FD 50 RT 90

CTRL-P

CTRL-P CTRL-A CTRL-K CTRL-P

CTRL-P CTRL-A CTRL-D S CTRL-L

f5 f1 f3

INST/DEL ← CTRL-N (n-r)

RT 35 INST/DEL (2 times) (n-r)

← (3 times) CTRL-G (n-r)

CTRL-B CTRL-F (n-r)

RT 45 FD 50 CTRL-A CTRL-O CTRL-L LT 45

SQUARE
```

3. You now need a procedure named SQUARE. Since it is a well known procedure, you will write this procedure using the TEACH file. By using TEACH, the picture that has been drawn will remain on the screen. Notice which CTRL characters can be used in TEACH and

which ones cannot be used in this mode. Circle those that work and cross out the ones that do not work in TEACH.

```
DRAW

TEACH
NAME:SQUARE

CTRL-G

TEACH
NAME:SQUARE

CTRL-N CTRL-B CTRL-O

f5 f3 f1 (n-r)

REAPEAT 4 [FD 50 RT 90] CTRL-A (n-r)

→ (2 times) CTRL-D CTRL-L

CTRL-P CTRL-A CTRL-K (n-r)

ST

CTRL-P INST/DEL (5 times) (n-r)

AND ← (2 times) INST/DEL (n-r)

E CTRL-C (n-r)

→ (2 times)

f3
```

4. Now you are working at toplevel again to experiment with commands necessary for drawing a CUBE. Type in the following commands and observe the results. Record your observations on the lines that are provided.

```
SQUARE
RT 45
FD 50
LT 45
```

```
SQUARE ______________________________
FD 50
LT 135
FD 50
LT 135
FD 50
LT 45
FD 50
RT 135
FD 50
RT 45
FD 50
RT 45
FD 50
RT 90 ______________________________
```

5. You are now going to define the procedure CUBE, illustrated in Figure 2-13. Please notice you will start using TEACH, make an intentional mistake after doing a considerable amount of typing, and be unable to go back and correct the error while in this mode. Therefore, you will prematurely end the procedure, switch to edit mode, and then complete writing the procedure. This lets you continue without using CTRL-G and starting all over.

   Enter the following commands and record the results on the lines provided.

```
TEACH
NAME:CUBE
SQUARE
RT 45
FD 50
LT 45
SQARE
← INST/DEL END ______________________
EDIT CUBE ______________________________
CTRL-N(5 times) CTRL-K(3 times) SQUARE
FD 50 __________________________________
LT 135
FD 50 __________________________________
```

```
RT 135 CTRL-A CTRL-D L ____________________
INST/DEL CTRL-L ____________________________
FD 50
LT 45
FD 50
RT 135
FD 50
FD 50
RT 45
FD 50
RT 50 ← (2 times) CTRL-D 9 → __________
f1 f3 f5 END (n-r) ________________________
CTRL-P (2 times)(n-r)
↑ (2 times)(n-r)
CTRL-A CTRL-O RT 45 (n-r) _________________
CTRL-N (2 times) (n-r)
↓ (3 times) (n-r)
CTRL-C (n-r)
CUBE ______________________________________
```

NOTE If this procedure does not resemble the picture in the illustration in Figure 2-13, compare it with Checkpoint 2-2, and make any necessary corrections to "debug" the procedure. Debug means to locate and correct errors in the procedure.

Turtle Talk. When using Commodore Logo, there are three basic modes or frameworks within which you can work.

Toplevel Mode:

This mode permits you to enter primitives or user-defined commands that execute immediately. These instructions must be reentered for each execution and do not contribute toward the development of a procedure.

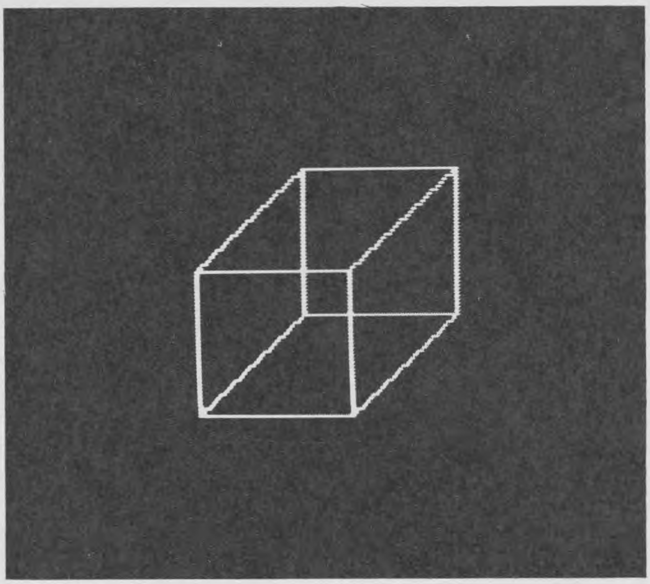


Figure 2-13.

```
TO CUBE
SQUARE
RT 45
FD 50
LT 45
SQUARE
FD 50
LT 135
FD 50
LT 135
FD 50
LT 45
FD 50
RT 135
FD 50
RT 45
FD 50
RT 45
FD 50
RT 90
END
```

Checkpoint 2-2.

TEACH Mode:

This mode is implemented by reading the TEACH file from the Utility Disk. After typing TEACH, the computer responds with NAME:. At this point, you name the procedure you wish to define. TEACH mode permits the user to enter Logo statements that sequentially define a procedure. When entering commands in this mode, the turtle does not draw. You are able to view any previously drawn material or recently typed text while you are using TEACH. The procedure is concluded by entering the word END and pressing the RETURN key. This results in a message reporting that your procedure is defined. The procedure is accessed simply by typing its name each time it is to be executed.

Edit Mode:

This mode is used to create or modify procedures. When attempting to modify a procedure, it is accessed by typing EDIT, followed by the procedure name (for example EDIT CUBE). When creating a new procedure or a series of procedures, it is accessed by typing TO, followed by the procedure name (for example TO CUBE). The screen display changes to show the procedure that you have chosen to edit. The line at the bottom of the screen informs you of the two options available for leaving the editor. All of the editing CTRL keys are available for use in modifying the procedure. To define the procedure and leave this mode, you must type CTRL-C. You will receive a message indicating that the procedure is defined. To abort the procedure, you must type CTRL-G and press return. You cannot view any drawing while in this mode.

Activity 2.6 reviews the various Logo control commands introduced in Activities 2.4 and 2.5, and lets you experiment with their use in three different Logo modes. You should have discovered that all CTRL commands are not available in every mode. Table 2-2 provides a summary of the availability of the control commands in each of the three operation modes.

Table 2-2. Summary of control commands available in the three operation modes.

	KEYS	TOP LEVEL	TEACH	EDIT
HORIZONTAL CONTROL	←	Y	Y	Y
	→	Y	Y	Y
	INST/DEL	Y	Y	Y
	CTRL-A	Y	Y	Y
	CTRL-L	Y	Y	Y

	KEYS	TOP LEVEL	TEACH	EDIT
VERTICAL CONTROL	CTRL-N	N	N	Y
	CTRL-P	N	N	Y*
	CTRL-O	N	N	Y
	↑	N	N	Y
	↓	N	N	Y
TEXT MANIPULATION	CTRL-D	Y	Y	Y
	CTRL-K	Y	Y	Y
	CTRL-P	Y**	N	N
SCREEN DISPLAY	f1	Y	Y	N
	f3	Y	Y	N
	f5	Y	Y	N
LEVEL CTRL	CTRL-G	Y	Y	Y
	CTRL-C	N	N	Y

*In this mode, CTRL-P moves to previous line.

**In this mode, CTRL-P yanks back the previously typed line or the most recently killed sequence of characters.

Turtle Truth. Examine the following commands and predict the results. Enter the commands to check your predictions. Compare your screen display to the illustrations in Figure 2-14.

```
DRAW
CUBE RT 90
CTRL-P
CTRL-P
CTRL-P
f5
f3
TEACH
```

```
NAME:4CUBE
f1
REPEAT 4 [CUBE RT 90]
HT
END
f3
DRAW
4CUBE
EDIT 4CUBE
→ (3 times) CTRL-D 6 CTRL-N (n-r)
CTRL-A CTRL-O FULLSCREEN
CTRL-K → (9 times) INST/DEL (n-r)
6 CTRL-L ← (3 times) CTRL-D 6 (n-r)
CTRL-C (n-r)
6CUBE
```

NOTE If this 6CUBE procedure does not resemble the illustration in Figure 2-14, compare it with Checkpoint 2-3 and make any necessary corrections to debug it.

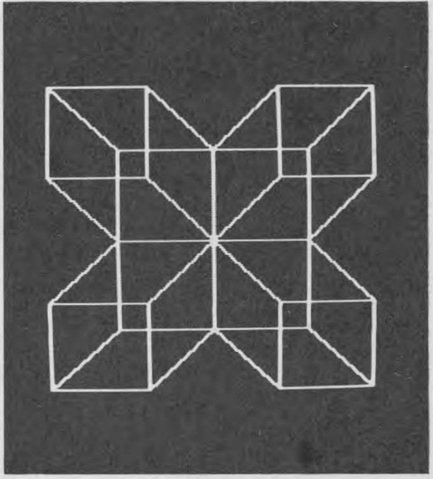


Figure 2-14a.

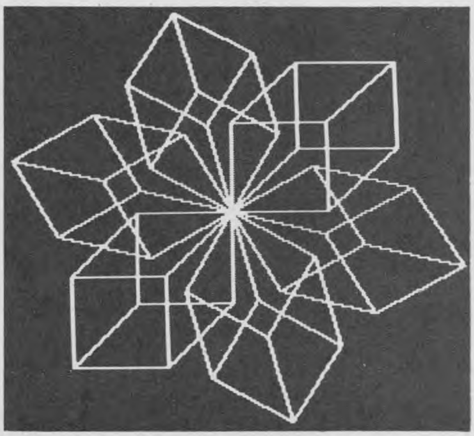


Figure 2-14b.

```
TO 6CUBE
FULLSCREEN
REPEAT 6 [CUBE RT 60]
HT
END
```

Checkpoint 2-3.

```
f1
POTS
```

NOTE The 4CUBE procedure still exists. By creating 6CUBE, you have merely modified the original procedure and defined it in another form.

Turtle Tester. Edit the procedure 6CUBE and define a new procedure named 8CUBE which is illustrated in Figure 2-15 and reads as follows:

```
TO 8CUBE
FULLSCREEN
REPEAT 8 [CUBE LT 45]
HT
END
```

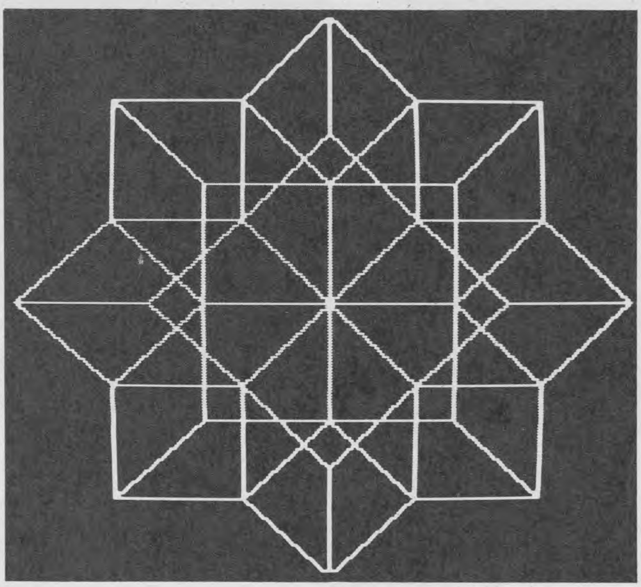


Figure 2-15.

Turtle Teaser. Edit one of the previous procedures to draw the design illustrated in Figure 2-16.

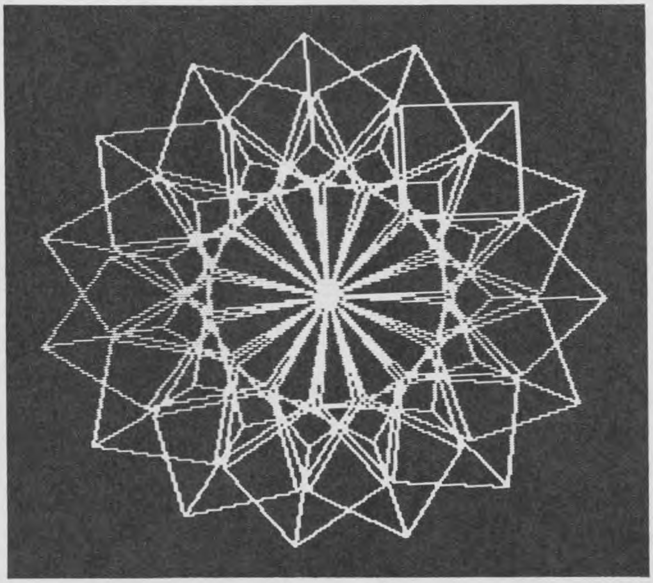


Figure 2-16.

NOTE Save all the procedures created in this activity under the file name 3.D.DESIGNS.

Reference Point 2

Command Review. Table 2-3 contains all of the CTRL keys introduced in the chapter. Each character is named, its function described, and its availability in each mode is indicated by an *. (TL stands for toplevel mode, TCH stands for TEACH mode, and EDIT stands for edit mode.)

Table 2-3. CTRL Keys introduced in Chapter 2.

Keys	Function	TL	TCH	EDIT
CTRL-A	Moves ■ to beginning of the line	*	*	*
CTRL-B	Moves ■ to the beginning of the edit buffer or one screenful of text backward			*
CTRL-C	Completes an edited procedure and returns you to toplevel			*
CTRL-D	Deletes character directly under ■	*	*	*
CTRL-F	Moves ■ one screenful of text forward or to the end of the edit buffer			*
CTRL-G	Stops an entry or procedure and returns you to previous point	*	*	*
CTRL-K	Erases everything on the current line to the right of ■	*	*	*
CTRL-L	Moves ■ to end of current line	*	*	*
CTRL-N or ↓	Moves ■ to next line of a procedure at current ■ position			*
CTRL-O	Opens a space for a new line directly above or before ■			*
CTRL-P	Inserts the line that was last "killed" by CTRL-K or previously typed	*	*	
CTRL-P	Moves ■ to previous line			*
CTRL-W	Stops the running of Logo until any other character is typed	*		
CTRL-Z	Stops the running of Logo and makes it pause until CO or CONTINUE is typed	*	*	
f1	Devotes entire screen to text	*		
f3	Splits the screen: top for graphics and bottom 5 lines for text	*		
f5	Devotes whole screen to graphics	*		
←	Moves ■ back one space and does not erase the character it moves over	*	*	*
→	Moves ■ ahead one space	*	*	*
CLR HOME	Scrolls the screen up to place current line in center of screen display			*

Keys	Function	TL	TCH	EDIT
INST DEL	Moves ■ back one space and erases characters it passes over	*	*	*

Application. Read ARCS and TEACH from the Utility Disk. Insert your file disk then enter the error-filled procedures listed below and illustrated in Figure 2-17a. Correct the procedures, using editing commands to produce the correct version also printed below and illustrated in Figure 2-17b. When the procedures work correctly, save them in a file named PICTURE.

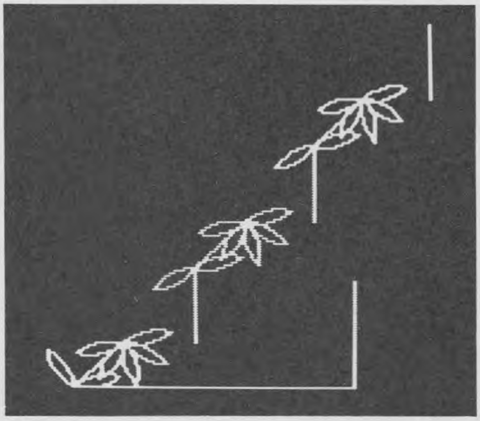


Figure 2-17a.

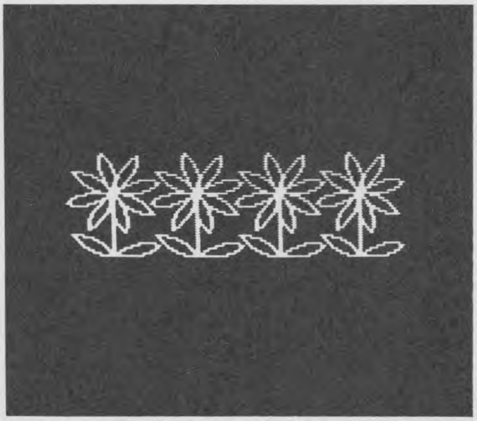


Figure 2-17b.

ERROR FILLED

```
TO GARDEN
BK 50
LT 90
FD 120
RT 90
PD
REPEAT 3 [FLOWER RT 90
 FD 35 LT 90]
END
```

CORRECT VERSION

```
TO GARDEN
PU
BK 50
LT 90
FD 120
RT 90
PD
REPEAT 4 [FLOWER
 PU RT 90 FD 35
 LT 90 PD]
```

```
TO FLOWER
LT 60
PETAL
SETH 0
RT 45
PETAL
FD 30
REPEAT 5 [PETAL
 RT 360/8]
PU
BK 30
PD
END

TO PETAL
ARCR 20 60
RT 120
ARCR 20 60
RT 120
END
```

```
PU HOME PD
END

TO FLOWER
LT 90
PETAL
SETH 0
RT 45
PETAL
SETH 0
FD 30
REPEAT 8 [PETAL
 RT 360/8]
PU
BK 30
PD
END

TO PETAL
REPEAT 2 [ARCR
 20 60 RT 120]
END
```

Suggested Projects

Go back into some of the previous files and alter the procedures using your new skill at editing. Save the new procedures you create under a different file name so that you still have the original file to use in future activities. For example:

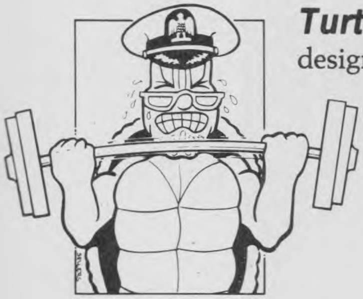
POLYGONS Change the length of the sides to make the shapes larger or smaller.

WABBIT.PART Change the formation of the facial features you added to the rabbit face.

Extending Turtle Trips

3

Activity 3.1 Turtle Input



Turtle Trials. This activity introduces the user to the design of procedures that include one input variable.

1. Enter the following commands and describe the screen display on the line provided.

```
DRAW
TO SQUARE
REPEAT 4 [FD 50 RT 90]
END
SQUARE ________________________________________
```

2. To modify the SQUARE procedure below use the command EDIT SQUARE. Don't forget to conclude the editing with CTRL-C.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END
```

Type the following and record your observations.

```
SQUARE ________________________________________
SQUARE :10 ____________________________________
SQUARE 10 _____________________________________
SQUARE 30 ________________________________________
SQUARE 50 ________________________________________
SQUARE 70 ________________________________________
```

NOTE SIDE is known as a variable. Variable names are identified by the prefix colon (:). A variable name is used to "hold" a numerical input value. This value is passed throughout a procedure by using the variable name. The reason for defining a procedure with a variable name is to gain flexibility. Above, the SQUARE procedure draws four different size squares by changing the value of the input variable SIDE.

3. Type in the following commands and record your observations where indicated.

```
f1
TO SQUARES _______________________________________
SQUARE 10
SQUARE 30
SQUARE 50
SQUARE 70
HT
END ______________________________________________

DRAW
SQUARES __________________________________________
```

4. Enter the commands as shown and record your observations on the lines. Compare your screen displays to Figure 3-1 and Figure 3-2.

```
DRAW
REPEAT 4 [SQUARES RT 360/4]
TO DESIGN :MANY
REPEAT :MANY [SQUARES RT 360/:MANY]
END

DRAW
DESIGN ___________________________________________
DESIGN 4 _________________________________________
DRAW
DESIGN 8 _________________________________________
```

```
f5
f3
```

NOTE A space is mandatory before the use of a variable following a primitive command and as an input variable for a procedure. A space is unnecessary when a variable is read as part of an arithmetic operation. Any unnecessary spaces are included for ease of readability.

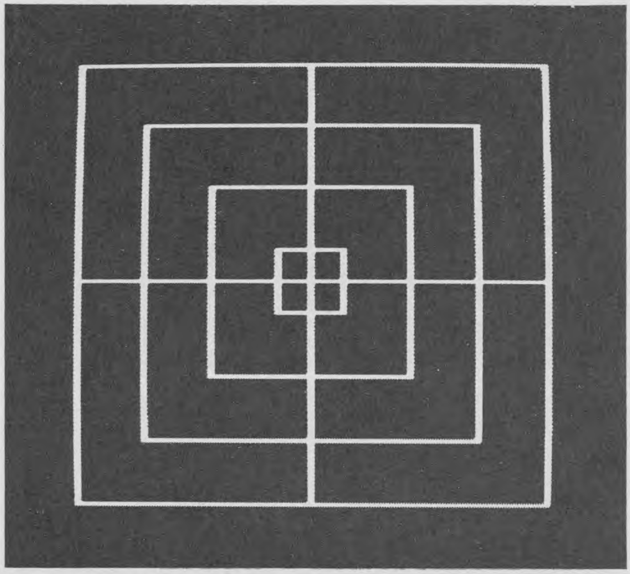


Figure 3-1.

5. Type in the following commands and describe what you see. Compare your screen displays with Figures 3-3, 3-4, and 3-5.

```
DRAW
ST
DESIGN 5  ____________________________________
f5
f3
DRAW
ST
DESIGN 9  ____________________________________
f5
f3
```

```
DRAW
DESIGN 2
```

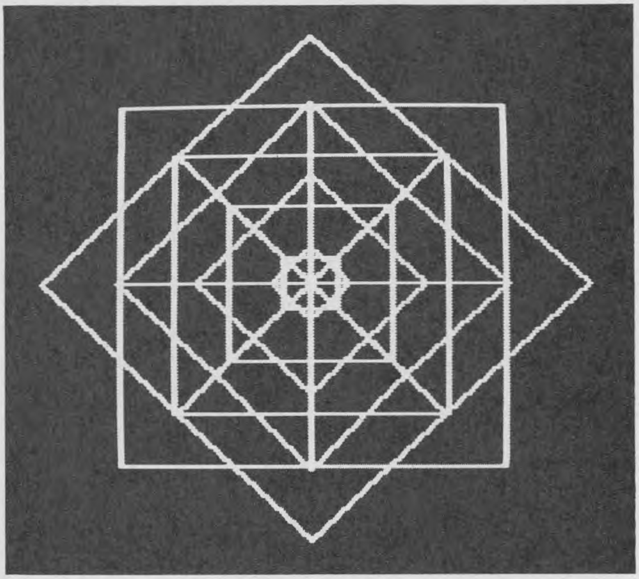


Figure 3-2.

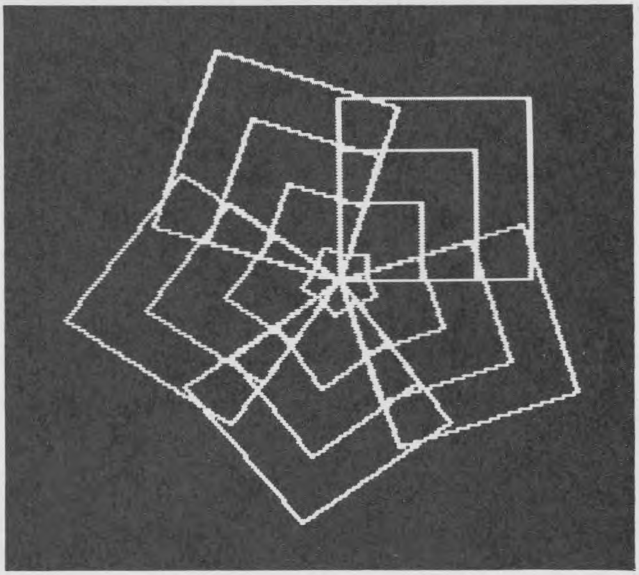


Figure 3-3.

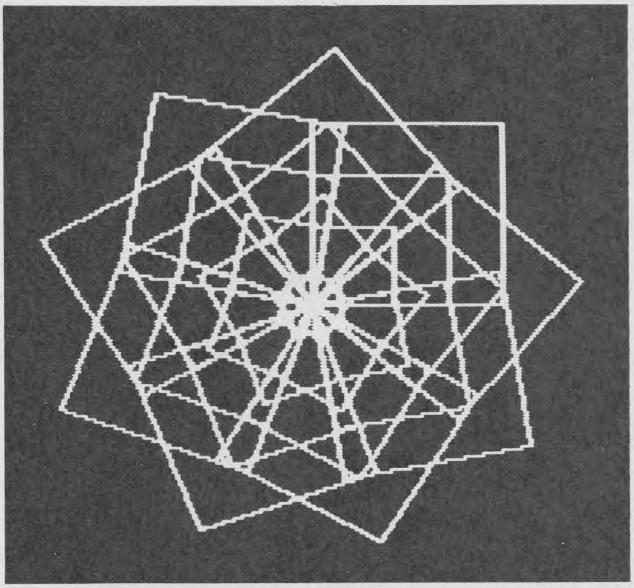


Figure 3-4.

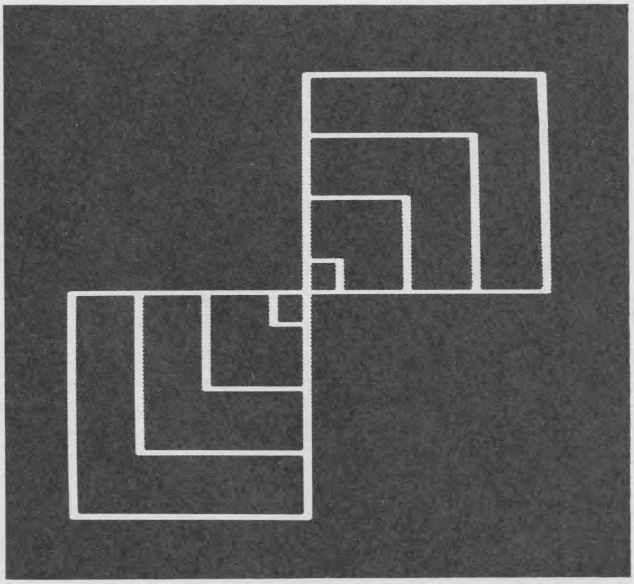


Figure 3-5.

6. Use the editor to modify the SQUARE and DESIGN procedures to read as follows:

```
TO SQUARE :S
REPEAT 4 [FD :S RT 90]
```

```
END

TO DESIGN :M
REPEAT :M [SQUARES RT 360/:M]
END

PO ALL
```

NOTE Use the screen display following PO ALL to verify your editing.

7. Enter the following commands and observe the results. Figure 3-6 displays DESIGN 12.

```
DESIGN 12
f5
DRAW
f3
ST
DESIGN M (Enter your own number for the input variable M)
```

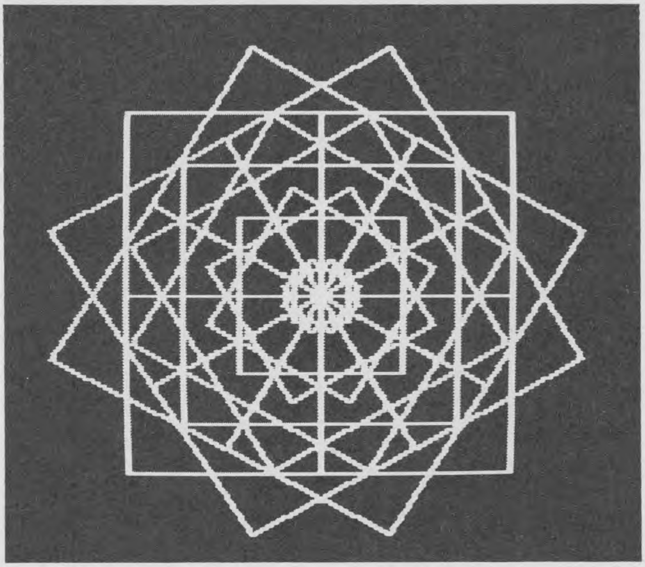


Figure 3-6.

Turtle Talk. One of the more powerful aspects of any programming language is the ability to use input variables. Part 1 introduces the concept of an input variable by contrasting two procedures:

```
TO SQUARE
REPEAT 4 [FD 50 RT 90]
END
```

and

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END
```

The previous activities have not used input variables. Instead, all commands have included constants. Constants are values that never change. SQUARE contains three constants: 4, 50, and 90. SQUARE is restricted to drawing a square with each side having a length of 50 units.

In contrast to a constant, an input variable can be considered to be a memory location that is identified by a name or group of characters (such as SIDE), and holds a value that can change. An input variable can be assigned any number. The procedure SQUARE :SIDE permits the square to be a size determined by providing SIDE with a numerical value. For example, SQUARE 10 draws a square with the length of each side being 10 units. SQUARE 70 draws a square with the length of each side being 70 units. Variables provide the programmer with much more versatility when defining procedures.

Part 4 defines a procedure that rotates square patterns. The DESIGN procedure includes a variable by the name of MANY to control the number of rotations. The expression REPEAT :MANY [SQUARES RT 360/:MANY] assures the completion of the TOTAL TURTLE TRIP. By including MANY in DESIGN, a variety of square rotational patterns can be displayed simply by entering commands such as DESIGN 4, DESIGN 9, or DESIGN N, where N is a value provided by the user.

Part 5 demonstrates that the name selected for the variable is not critical. The replacement of S for SIDE and M for MANY does not alter the designs produced by the procedures. One-letter variable names are more convenient for the less skilled typist, but are not as meaningful for other readers (or the program author at a later time) and are to be discouraged. Full mnemonic names are used throughout the remainder of the book. The reader may substitute abbreviations if so desired.

Activity 3.1 introduces you to the use of one variable when defining procedures. Future activities will reinforce and extend your ability to incorporate more than one variable into procedures.

Turtle Truth. Study the following procedure and project its outcome. Enter the commands and check your predictions. Compare the screen display to the pictures illustrated in Figures 3-7, 3-8, 3-9, and 3-10.

```
DRAW
TO TRIANGLE :SIDE
REPEAT 3 [FD :SIDE RT 120]
END

TRIANGLE 40 ______________________________
TRIANGLE 60 ______________________________
TRIANGLE 80 ______________________________
TRIANGLE 90 ______________________________
TO TRIANGLES
TRIANGLE 40
TRIANGLE 60
TRIANGLE 80
TRIANGLE 90
END

TO TRIDESIGN :MANY
FULLSCREEN
REPEAT :MANY [TRIANGLES RT 360/:MANY]
END

DRAW
```

```
TRIDESIGN 2 ______________________________
f3
DRAW
TRIDESIGN 6 ______________________________
f3
DRAW
TRIDESIGN 9 ______________________________
f3
DRAW
TRIDESIGN 17 _____________________________
f3
DRAW _____________________________________
```

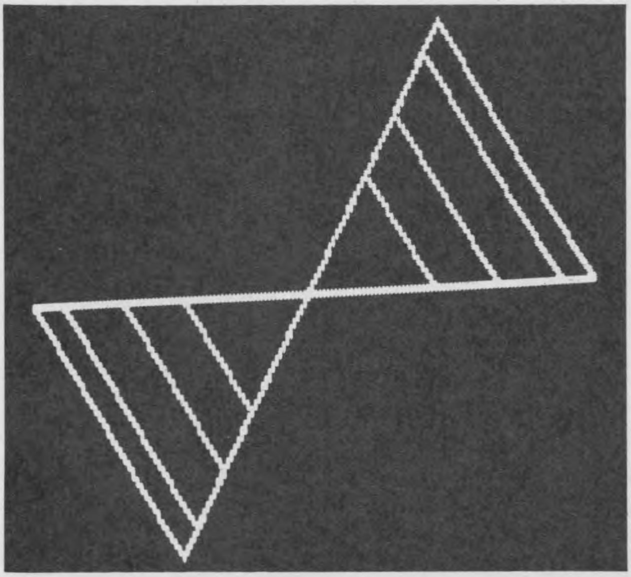


Figure 3-7.

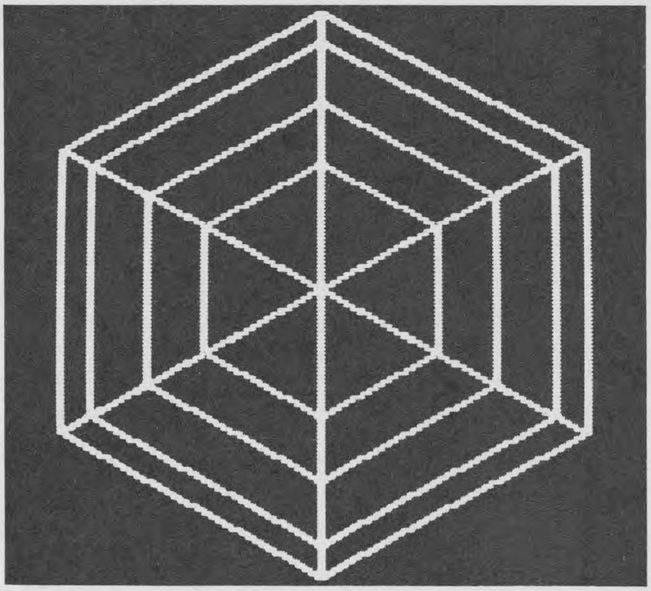


Figure 3-8.

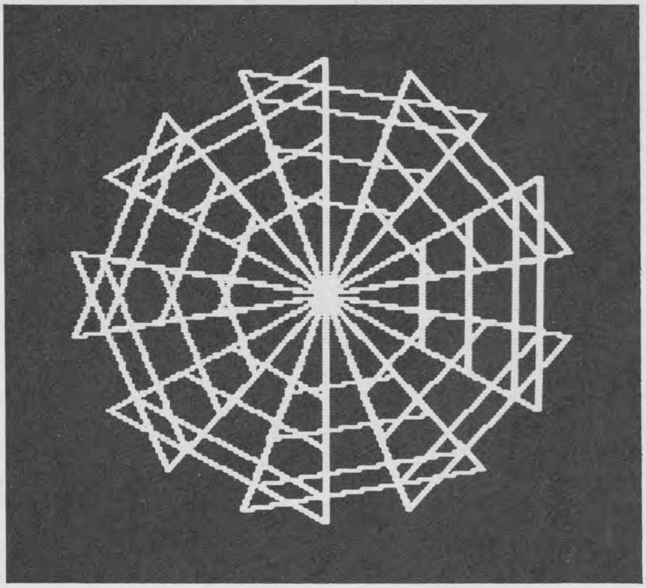


Figure 3-9.

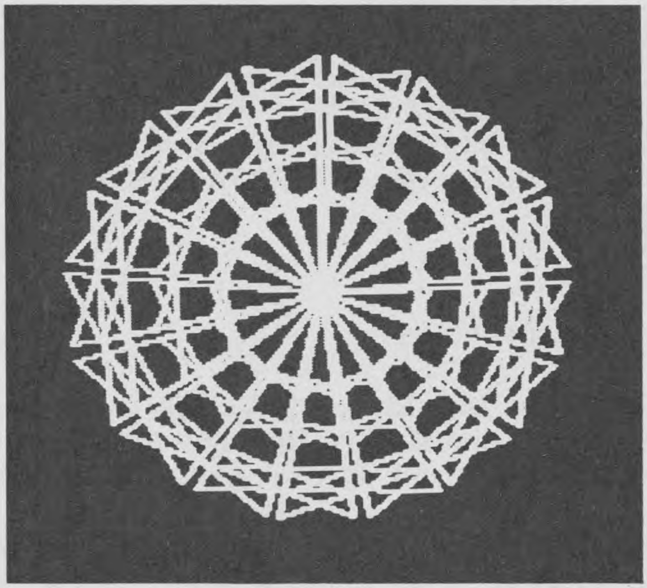


Figure 3-10.

NOTE Save these procedures under the file name VAR.ROTATIONS.

Turtle Tester. Erase the procedures in the workspace with GOODBYE. Define the new procedures suggested below and save them in a file named VAR.ROTATIONS2. (Answer on page 290.)

1. Define a procedure called SQUARES.2. In the procedure, include different inputs for the variable SIDE in SQUARE.
2. Define a procedure named SQDESIGN.2, which uses the variable MANY and the newly defined procedure SQUARES.2. Experiment using different inputs for MANY and observe the resulting changes in the displays.

Turtle Teaser. Erase the workspace. Load the file named VAR.ROTATIONS. Study the illustration in Figure 3-11. Using the existing procedures, define a new procedure named COMBO that duplicates the illustration. When this is accomplished, type the following to add COMBO to your VAR.ROTATIONS file.

```
SAVE "VAR.ROTATIONS
```

NOTE Commodore Logo lets you SAVE a file by the same name as one that already exists on disk. The disk drive will not duplicate a file name. Logo will instruct the disk drive to replace the existing file with the new file. Thus, the information in the original file will be lost. It is advisable to CATALOG before using the SAVE command to avoid an accidental duplication of file names.

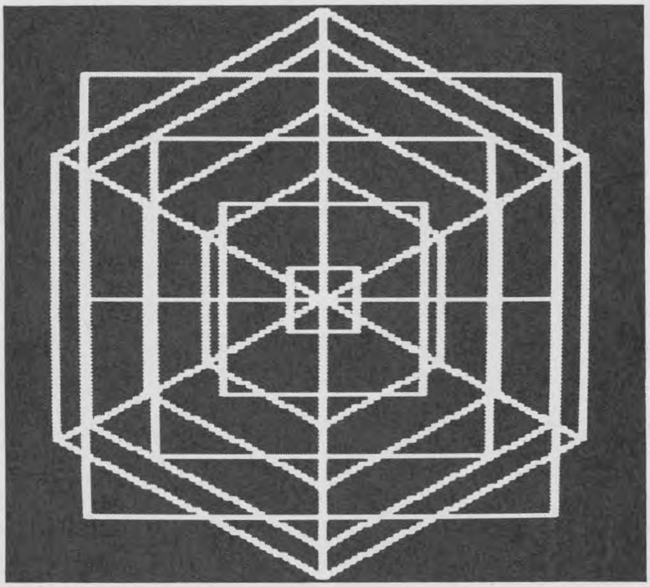
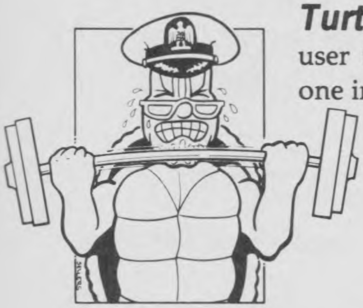


Figure 3-11.

Activity 3.2 Turtle Hues



Turtle Trials. This activity uses color to introduce the user to the design of procedures that include more than one input variable.

1. Enter the following commands and describe the screen display on the lines provided.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

SQUARE ________________________________
SQUARE 20 _____________________________
CS
SQUARE 40 _____________________________
```

2. Use the editor to modify the SQUARE procedure to read as follows:

(Edit using the command, EDIT "SQUARE)

```
TO SQUARE :SIDE
PENCOLOR 2
REPEAT 4 [FD :SIDE RT 90]
END

SQUARE 30 _____________________________
```

3. Use the editor to alter the SQUARE procedure to read as follows:

```
TO SQUARE :SIDE :COLOR
PENCOLOR :COLOR
REPEAT 4 [FD :SIDE RT 90]
END
```

```
SQUARE 40 ________________________________
SQUARE 40 :3 _____________________________
SQUARE 40 3 ______________________________
SQUARE 20 5 ______________________________
SQUARE 70 14 _____________________________
SQUARE 60 15 _____________________________
SQUARE 60 11 _____________________________
```

4. Enter the commands and describe the screen display on the lines provided.

```
DRAW
TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90]
END
RECTANGLE 10 40 __________________________
PENCOLOR 1
RECTANGLE 10 40 __________________________
DRAW
PC 10
RECTANGLE 30 30 __________________________
DRAW
PC 8
RECTANGLE 5 60 ___________________________
DRAW
PC 14
RECTANGLE 20 25 __________________________
```

5. Use the editor to modify the RECTANGLE procedure to read as follows:

```
TO RECTANGLE :LENGTH :WIDTH :COLOR
PC :COLOR
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90]
END

RECTANGLE 20 40 __________________________
RECTANGLE 20 40 4 ________________________
DOUBLECOLOR ______________________________
```

```
RECTANGLE 60 10 13 ____________________________
RECTANGLE 40 40 0 _____________________________
SINGLECOLOR __________________________________
RECTANGLE 80 50 9 _____________________________
RECTANGLE 100 80 6 ____________________________
DRAW

SAVE "COLOR.SHAPE
ERASE ALL ____________________________________
```

6. Modify the RECTANGLE procedure to read as follows using the editor.

```
RECTANGLE :LENGTH :WIDTH :COLOR
DOUBLECOLOR
PC :COLOR
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90]
END

RECTANGLE 70 100 7 ____________________________
```

How does the thickness of these lines compare with those from Part 5?

____

```
RECTANGLE 100 60 8 ____________________________
```

How many rectangles appear on the screen? ______________________

Turtle Talk. Procedures can be defined to include more than one input variable. In Part 1, the variable SIDE is used in the procedure SQUARE to define the length of each side. Color is introduced in Part 2 with the PENCOLOR (abbreviated PC) command. PC N, where N = 0, 1, 2, 3, 4, or 5 designates the color for the diagram to be drawn. Table 3-1 displays the color each number represents when drawn on a gray (11) background screen.

SQUARE is modified in Part 3 to include two variables (SIDE representing the length of each side, and COLOR designating the drawing color). Now SQUARE may be drawn to any size and in any color. Simply enter SQUARE followed by two input variables, separated by a space. Notice in Part 3, SQUARE 60 11 causes the square to disappear since PENCOLOR 11 is the color of the background screen.

Table 3-1. Color representations on a black screen.

Color	Number
Black	0
White	1
Red	2
Cyan	3
Purple	4
Green	5
Blue	6
Yellow	7
Orange	8
Brown	9
Light Red	10
Gray 1	11
Gray 2	12
Light Green	13
Light Blue	14
Gray 3	15

The use of these variables in the definition of a procedure is illustrated in Part 5. The various sizes and colors of rectangles drawn in Part 5 demonstrate the possible selection of hues available in Logo. The versatility and ease of using variables to define procedures is evident. Part 5 also introduces the SINGLECOLOR and DOUBLECOLOR commands. Using DOUBLECOLOR draws all turtle paths with thicker lines. When Logo is booted, it defaults to SINGLECOLOR mode. Switching back and forth between SINGLECOLOR and DOUBLECOLOR clears the graphics screen. Part 6 demonstrates that DOUBLECOLOR can be included as a statement within a procedure.

Turtle Truth. Study the following commands and predict the results. Enter the commands to confirm your prediction. (The file PICTURE was created in the Reference Point for Chapter 2.)

```
READ "ARCS
READ "PICTURE
```

NOTE Remember, ARCS is located on the Logo Utilities Disk.

```
TO STEM :COLORSTEM
PC :COLORSTEM
LT 90
LEAF
RT 90
LEAF
SETH 0
FD 30
END

TO LEAF
REPEAT 2 [ARCR 23 60 RT 120]
END
```

Use the editor to modify the FLOWER and GARDEN procedures to read as follows:

```
TO FLOWER :COLORFLOWER
PC :COLORFLOWER
REPEAT 8 [PETAL RT 360/8]
PU
BK 30
PD
END

TO GARDEN :COLORSTEM :COLORFLOWER
PU
BK 50
LT 90
FD 120
RT 90
PD
REPEAT 4 [STEM :COLORSTEM FLOWER
  :COLORFLOWER PU RT 90 FD 35 LT 90 PD]
PU HOME PD
END

GARDEN 5 2 ______________________________
DRAW
GARDEN 13 4 _____________________________
```

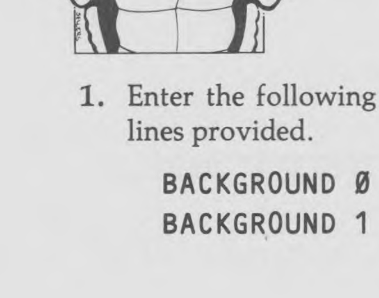
```
DRAW
GARDEN 14 7  ______________________________
```

NOTE Save these procedures in a file named PICTURE2.

Turtle Tester. Write a program that uses two variables and creates a differently shaped flower in several colors. Save these procedures under a name of your choice.

Turtle Teaser. Write a program that draws a row of flowers. Alternate the drawing of the flower created in Turtle Truth with the one created in Turtle Tester.

Activity 3.3 Turtle Background



Turtle Trials. This activity demonstrates the possibilities for drawing on a non-black background screen and introduces a method for displaying messages on the screen.

1. Enter the following commands and describe the screen display on the lines provided.

```
BACKGROUND 0  ______________________________
BACKGROUND 1  ______________________________
```

```
BG 2 ____________________________________
BG 3 ____________________________________
BG 4 ____________________________________
BG 5 ____________________________________
BG 6 ____________________________________
BG 7 ____________________________________
BG 8 ____________________________________

TO FLASH
BG 9
BG 10
BG 11
BG 12
BG 13
BG 14
BG 15
END

FLASH ___________________________________
```

2. Use the editor to modify the FLASH procedure and define WAIT to read as follows:

```
TO FLASH2 :TIME
BG 9
WAIT :TIME
BG 10
WAIT :TIME
BG 11
WAIT :TIME
BG 12
WAIT :TIME
BG 13
WAIT :TIME
BG 14
WAIT :TIME
BG 15
WAIT :TIME
END
```

```
TO WAIT :TIME
REPEAT :TIME*500 []
END
FLASH2 1 ________________________________
FLASH2 2 ________________________________
FLASH2 3 ________________________________
```

3. Modify the FLASH2 procedure to read as follows and enter the new MESSAGE procedure. Observe and record the results of the screen displays.

```
TO FLASH3 :TIME
CLEARTEXT
BG 9
MESSAGE :TIME
BG 10
MESSAGE :TIME
BG 11
MESSAGE :TIME
BG 12
MESSAGE :TIME
BG 13
MESSAGE :TIME
BG 14
MESSAGE :TIME
BG 15
MESSAGE :TIME
BG 12
PRINT [GRAY IS THE BACKGROUND COLOR]
PRINT [WHEN YOU START WITH LOGO]
WAIT :TIME + 2
CLEARTEXT
PRINT [THE PENCOLOR IS WHITE]
PRINT []
PRINT []
PRINT [WHEN YOU START WITH LOGO]
WAIT :TIME + 1
CLEARTEXT
```

```
END

TO MESSAGE :TIME
PRINT [A NEW BACKGROUND COLOR]
WAIT :TIME
CLEARTEXT
END

FLASH3 2 ______________________________
FLASH3 3 ______________________________
FLASH3 4 ______________________________
```

4. Enter the following commands and observe the results. Record your observations on the lines provided.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO TRIANGLE :SIDE
REPEAT 3 [FD :SIDE LT 120]
END
TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH LT 90 FD :WIDTH LT 90]
END

TO HOUSE :SIDE
ST
SQUARE :SIDE
FD :SIDE RT 90
TRIANGLE :SIDE
FD :SIDE/2
PU RT 90 FD :SIDE/2 PD
RECTANGLE :SIDE/2 :SIDE/3
HT
END

DRAW
BG 1 ___________________________________
```

```
PC 0 ________________________________________
HOUSE 50 ____________________________________
DRAW
BG 6 ________________________________________
PC 8 ________________________________________
HOUSE 60 ____________________________________
DRAW
BG 11 _______________________________________
PC 10 _______________________________________
HOUSE 20 ____________________________________
```

5. Enter the following commands and observe the effect of each PenColor as it draws on each BackGround color. Describe the color of the lines in each drawing.

```
TO HOUSEDEMO :SIDE :TIME
HOUSE :SIDE
WAIT :TIME
DRAW
END

TO COLORDEMO :COLOR :SIDE
CLEARTEXT
PC :COLOR
BG 1
PRINT [PENCOLOR IS ] PRINT :COLOR
HOUSEDEMO :SIDE 6
BG 5
HOUSEDEMO :SIDE 6
BG 4
HOUSEDEMO :SIDE 6
BG 8
HOUSEDEMO :SIDE 6
BG 6
HOUSEDEMO :SIDE 6
BG 0
HOUSEDEMO :SIDE 6
CLEARTEXT
END
```

```
COLORDEMO 1 65
     white PC on white BG  ________________
              on green BG  ________________
              on purple BG  _______________
              on orange BG  _______________
              on blue BG  _________________
              on black BG  ________________

COLORDEMO 2 20
     red PC on white BG  __________________
              on green BG  ________________
              on purple BG  _______________
              on orange BG  _______________
              on blue BG  _________________
              on black BG  ________________

DOUBLECOLOR
COLORDEMO 2 20
     red PC on white BG  _________________
              on green BG  ________________
              on purple BG  _______________
              on orange BG  _______________
              on blue BG  _________________
              on black BG  ________________

SINGLECOLOR
COLORDEMO 9 30
     brown PC on white BG  ________________
               on green BG  _______________
               on purple BG  ______________
               on orange BG  ______________
               on blue BG  ________________
               on black BG  _______________

DOUBLECOLOR
COLORDEMO 9 30
     brown PC on white BG  ________________
               on green BG  _______________
               on purple BG  _______________
               on orange BG  _______________
               on blue BG  _________________
               on black BG  ________________

SINGLECOLOR
COLORDEMO 3 40
     cyan PC on white BG  _________________
               on green BG  _______________
               on purple BG  ______________
               on orange BG  ______________
               on blue BG  ________________
               on black BG  _______________

COLORDEMO 7 50
     yellow PC on white BG  _______________
            on green BG  __________________
            on purple BG  _________________
            on orange BG  _________________
            on blue BG  ___________________
            on black BG  __________________

COLORDEMO 0 60
     black PC on white BG  ________________
              on green BG  ________________
              on purple BG  _______________
              on orange BG  _______________
              on blue BG  _________________
              on black BG  ________________
```

6. Modify the MESSAGE procedure and enter the two new procedures. Then type in the commands and observe the results. Notice that an object first drawn in color on a black screen changes peculiarly as the background color is altered.

```
TO MESSAGE2
PRINT [A NEW PENCOLOR]
END

TO BGFLASH :SIDE :TIME
```

```
HOUSE :SIDE
MESSAGE2
WAIT :TIME
FLASH2 :TIME
CLEARTEXT
DRAW
END

TO FLASHDEMO :SIDE :TIME
PC 1
BGFLASH :SIDE :TIME
PC 2
BGFLASH :SIDE :TIME
PC 3
BGFLASH :SIDE :TIME
PC 4
BGFLASH :SIDE :TIME
PC 5
BGFLASH :SIDE :TIME
PC 0
BGFLASH :SIDE :TIME
END

FLASHDEMO 60 20
```

Turtle Talk. Logo allows for the drawing of diagrams on a colored background screen by using the BG (abbreviation for BackGround) command followed by a numerical input from 0 to 15. Table 3-1 summarizes the available colors with their corresponding number codes. Input 11 is the color of the background when the computer is booted with Logo.

In Part 2, the WAIT procedure is developed to hold a screen display and cause a pause in the running of a procedure. The command is written WAIT :TIME, where TIME is an input number. TIME is given several values during the execution of the activity to demonstrate the different length of pauses.

It is possible to display text and a graphics screen by using the PRINT or PR command. The five bottom lines are available for text display, although the fifth line usually contains the cursor. The message is displayed by entering PRINT followed by the text enclosed within the square brackets ([]). The line PRINT [] is used to create a space between printed messages to improve the appearance of the display.

Notice in Parts 1, 2, and 3, the procedure FLASH is modified several times. The originals are left intact so they may be used in their initial forms. Each modification is assigned a different name by appending a number, then is newly defined by typing CTRL-C.

Part 4 involves a sophisticated use of variables within a procedure. Each sub-procedure is assigned a variable in proportion to the controlling variable entered when calling the main procedure. In this way, the procedure HOUSE :SIDE is able to draw a house of any size with all parts in a consistent proportion.

Part 5 demonstrates various size houses drawn in several PenColors on a variety of BackGround colors. These demos display the limitations of drawing colored lines on differently colored backgrounds. Your observations and notes during these parts should indicate that black and white pens draw on any background other than themselves. However, there are some peculiar results with other combinations. The pencolor is sometimes a different color than expected. Also some lines are thicker than others. Drawing in DOUBLECOLOR, rather than SINGLECOLOR, provides some improvement in the appearance of the display. The reader must experiment with mixing various background colors with pencolors to obtain a pleasing combination. It should be mentioned that these strange effects occur because of the Commodore computer system and not because of Logo.

Turtle Truth. Examine the following commands and predict the results. Enter the commands to check your predictions and compare the final screen display with Figure 3-12.

```
TO PANES :SIDE
SQUARE :SIDE/4
SQUARE :SIDE/8
FD :SIDE/8 RT 90
FD :SIDE/8 LT 90
SQUARE :SIDE/8
LT 90 FD :SIDE/8
RT 90 BK :SIDE/8
END
```

Modify the House procedure to read as follows:

```
TO HOUSE2 :SIDE :COLORPANES :COLORSQUARE
  :COLORTRIANGLE :COLORRECTANGLE
ST
PC :COLORSQUARE
SQUARE :SIDE
FD :SIDE RT 90
PC :COLORTRIANGLE
TRIANGLE :SIDE
FD :SIDE/2 PU RT 90 FD :SIDE/2 PD
PC :COLORRECTANGLE
RECTANGLE :SIDE/2 :SIDE/3
PU RT 90 FD :SIDE/10 LT 90 PD
PC :COLORPANES
PANES :SIDE
PU BK :SIDE/3 PD
PANES :SIDE
PU LT 90 FD :SIDE/2 RT 90 PD
PANES :SIDE
PU FD :SIDE/3 + :SIDE/2 RT 90 FD :SIDE/10 + :SIDE/
  3 + :SIDE/2 RT 90 PD
HT
END

TO NEIGHBORHOOD :TIME
PU LT 90 FD 50 RT 90 PD
HOUSE2 30 7 5 13 1
PRINT [ONE SMALL HOUSE IS ON THE STREET.]
PU RT 90 FD 50 LT 90 PD
HOUSE2 45 4 14 6 8
PR [SOMEONE BUILDS NEXT DOOR.]
PU RT 90 FD 65 LT 90 PD
HOUSE2 60 10 12 3 8
PR [NOW IT IS A NEIGHBORHOOD.]
PRINT []
PR [WATCH THE STRANGE COLOR EFFECTS!]
WAIT :TIME
FLASH2 :TIME
CLEARTEXT
DRAW
END
```

```
    NEIGHBORHOOD 10
```

NOTE Save these procedures in a file named COLORFUL.STUFF.

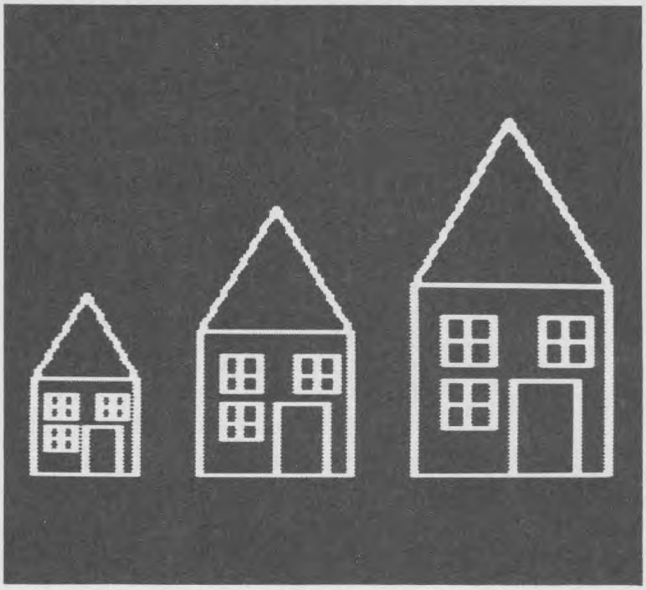
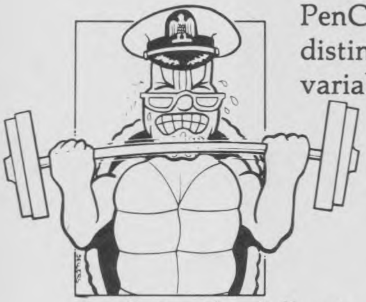


Figure 3-12.

Turtle Tester. Erase the current workspace and create a new file of procedures to draw a garage. Use variables for the size and color so the garage can be drawn in a variety of sizes and colors. Draw the garage on several background colors. Save the file under a name of your choice.

Turtle Teaser. Add the garages to the neighborhood drawn in Turtle Truth.

Activity 3.4 Turtle Variables



Turtle Trials. This activity further explores the use of PenColor and BackGround color. It also introduces and distinguishes the terms "local variables" and "global variables."

1. Enter the following commands to load a previously created file from your file disk. Then add the following procedures to the file and observe the results. Describe your observations on the lines provided.

```
READ "PICTURE2
POTS

TO POT
ST
RT 90 FD 15 RT 90
REPEAT 2 [FD 10 RT 90 FD 30 RT 90]
FD 10 RT 90 FD 5
REPEAT 4 [FD 20 LT 90]
BK 5 RT 90 FD 10 LT 90 FD 15 RT 90
HT
END

POT 1   ________________________________
PENERASE
PC 13
POT 1   ________________________________
PC -1
POT 1   ________________________________
ST
```

2. Enter the following procedure and observe the different results when using PENERASE (PC -1) on colored backgrounds. Describe the screen displays on the lines provided.

```
TO PE.DEMO :COLORBG
BG :COLORBG
POT
END

PE.DEMO 2  ______________________________
PC -1
PE.DEMO 2  ______________________________
PC 1
PE.DEMO 5  ______________________________
PC -1
PE.DEMO 5  ______________________________
PC 7
PE.DEMO 4  ______________________________
PC -1
PE.DEMO 4  ______________________________
PC 1
```

3. Edit POT and enter PLANT to read as follows. Compare your PLANT with the one illustrated in Figure 3-13. Observe the screen display and record your observations on the lines provided.

```
TO POT :COLORPOT
ST
PC :COLORPOT
RT 90 FD 15 RT 90
REPEAT 2 [FD 10 RT 90 FD 30 RT 90]
FD 10 RT 90 FD 5
REPEAT 4 [FD 20 LT 90]
BK 5 RT 90 FD 10 LT 90 FD 15 RT 90
HT
END

TO PLANT :COLORPOT :COLORSTEM :COLORFLOWER
POT :COLORPOT
STEM :COLORSTEM
FLOWER :COLORFLOWER
END
```

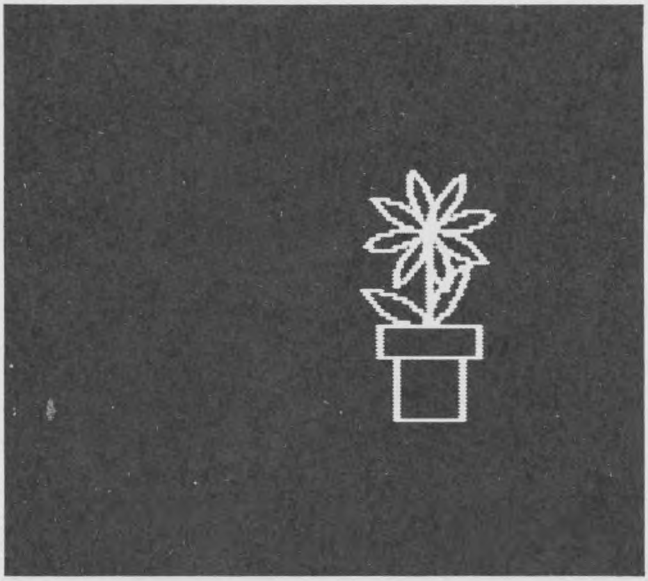


Figure 3-13.

```
PLANT 9 5 7  ______________________________
BG 1
DRAW
PLANT 10 4 3 ______________________________
BG 7
DRAW
PLANT 0 14 8  _____________________________
```

4. Enter the following procedures and observe the results. Compare your screen display to the illustration in Figure 3-14.

```
TO POSIE :COLORSTEM :COLORFLOWER
STEM :COLORSTEM
FLOWER :COLORFLOWER
END

POSIE 1 10  _______________________________

TO VARIABLEDEMO
BG :COLORSTEM
PC :COLORFLOWER
END
```

```
VARIABLEDEMO ________________________________
```

NOTE Do not be alarmed at the error message. COLORSTEM and COLORFLOWER are undefined in VARIABLEDEMO even though they are defined in POSIE.

5. Enter the following procedure and describe the results.

```
TO ROW :COLORSTEM :COLORFLOWER
REPEAT 7 [POSIE :COLORSTEM :COLORFLOWER PU
  RT 90 FD 40 LT 90 PD]
END

ROW 5 7 ________________________________
```

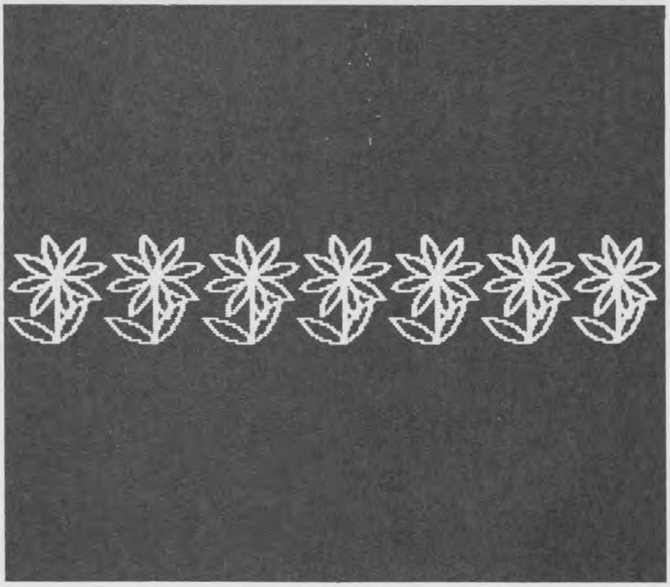


Figure 3-14.

6. Modify the procedure POSIE and describe the screen display on the lines provided.

```
TO POSIE
MAKE "COLORSTEM 9
MAKE "COLORFLOWER 14
STEM :COLORSTEM
```

```
FLOWER :COLORFLOWER
END
```

```
POSIE ____________________________________
VARIABLEDEMO ______________________________
```

What color is the background screen? ______________________________

What color is the turtle? ______________________________

NOTE You should observe that the background screen changed to the color of COLORSTEM and the turtle changed to the color of COLORFLOWER. Notice the COLORSTEM and COLORFLOWER variables are now defined in VARIABLEDEMO.

7. Enter the procedure ASSIGN and modify POSIE, STEM, FLOWER, and ROW as indicated below. Describe the screen display on the lines provided.

```
TO ASSIGN
MAKE "COLORSTEM 9
MAKE "COLORFLOWER 14
END

TO POSIE
STEM
FLOWER
END

TO STEM
PC :COLORSTEM
LT 90
LEAF
RT 90
LEAF
SETH 0
FD 30
END

TO FLOWER
PC :COLORFLOWER
```

```
REPEAT 8 [PETAL RT 360/8]
PU
BK 30
PD
END

TO ROW
ASSIGN
REPEAT 7 [POSIE PU RT 90 FD 40 LT 90 PD]
END

ROW ______________________________________
```

8. Modify ASSIGN and ROW as indicated below. Describe the screen display.

```
TO ASSIGN
MAKE "COLORSTEM 9
MAKE "COLORFLOWER RANDOM 16
END

TO ROW
REPEAT 7 [ASSIGN POSIE PU RT 90 FD 40 LT 90 PD]
END

ROW ______________________________________
```

9. When COLORFLOWER in Part 8 is assigned the color RANDOM 16, it can take on any value from 0 to 15 and display that color. It is possible that 11 could be selected and a gray flower would be drawn on a gray background. In order to avoid this possibility, make the following change in the ASSIGN procedure to prevent gray (PC 11) from being randomly selected. Describe the screen display.

```
TO ASSIGN
MAKE "COLORSTEM 9
MAKE "COLORFLOWER RANDOM 11
END
```

```
ROW  ________________________________________

SAVE "COLORFUL.POSIES
```

Turtle Talk. As a review, recall that PENDOWN (PD) and PENUP (PU) establish conditions for allowing the turtle to draw (PD) or not to draw (PU). This activity demonstrates how PENERASE (abbreviated PC-1) retraces an existing diagram and erases it.

Part 1 demonstrates that PENERASE erases any previously drawn diagrams encountered in the turtle's path. Part 2 illustrates that the same erasing occurs for different colored backgrounds and pens.

In Logo, input variables consist of two types: local variables and global variables. All input variables that appear in Activities 3.1, 3.2, and 3.3 are classified as local variables.

Local variables are defined by prefixing a variable name with a colon (:) when defining a procedure. For example, in Part 4, TO POSIE :COLORSTEM :COLORFLOWER defines COLORSTEM and COLORFLOWER to be local variables.

An important characteristic of local variables is that they are defined only within the procedure that references them. Thus, COLORSTEM and COLORFLOWER are local variables for POSIE. In Part 4, VARIABLEDEMO demonstrates that COLORSTEM and COLORFLOWER are undefined, since they are not local variables for that procedure. Part 5 defines COLORSTEM and COLORFLOWER to be local variables for ROW as well as POSIE.

In contrast to local variables, global variables are defined by using the MAKE command. For example, in Part 6, MAKE "COLORSTEM 9 defines COLORSTEM to be a global variable that receives the value 9. Global variables differ from local variables by being defined outside of procedures that reference them as well as in the procedure that defines the variable. In Part 6, VARIABLEDEMO demonstrates that COLORSTEM and COLORFLOWER have the values that are defined in POSIE, even though they are not explicitly defined in VARIABLEDEMO.

In Part 8, the ASSIGN procedure gives global variables COLORSTEM and COLORFLOWER respective values of 9 and 14. Since the revised version of ROW now includes ASSIGN, the COLORSTEM and COLORFLOWER variables in POSIE are defined globally. Therefore, the inclusion of COLORSTEM and COLORFLOWER when defining the respective STEM and FLOWER procedures is unnecessary.

The RANDOM command in Part 8 permits an easy way to generate random numbers. RANDOM N generates whole numbers, inclusively, from 0 to N-1. For example, RANDOM 16 generates numbers of the form 0, 1, 2, . . . , 15. Since 11 is gray, the random selection of a non-gray color can be obtained by RANDOM 11, eliminating the possibility of selecting 11. It generates numbers 1, 2, 3, . . . , 10, and does not permit the selection of numbers 12, 13, 14, or 15. Activity 3.7 will introduce a method by which only the number 11 will be eliminated from the random selection.

Turtle Truth. Examine the following procedures and predict the resulting screen display. Enter the commands to verify your predictions and compare the final screen display to the picture illustrated in Figure 3-15.

```
TO PETALS :ARCLENGTH
MAKE "COLOR RANDOM 11
PC :COLOR
REPEAT 2 [ARCR :ARCLENGTH 60 RT 120]
END

TO ANYFLOWER :ARCLENGTH :MANY
REPEAT :MANY [PETALS :ARCLENGTH RT 360/:MANY]
END

TO DAISY
ANYFLOWER 50 16
DRAW
ANYFLOWER 50 8
ANYFLOWER 35 8
REPEAT 20 [BG RANDOM 11 WAIT 3]
BG 15
END

TO WAIT :TIME
REPEAT :TIME*500 []
END
```

DAISY

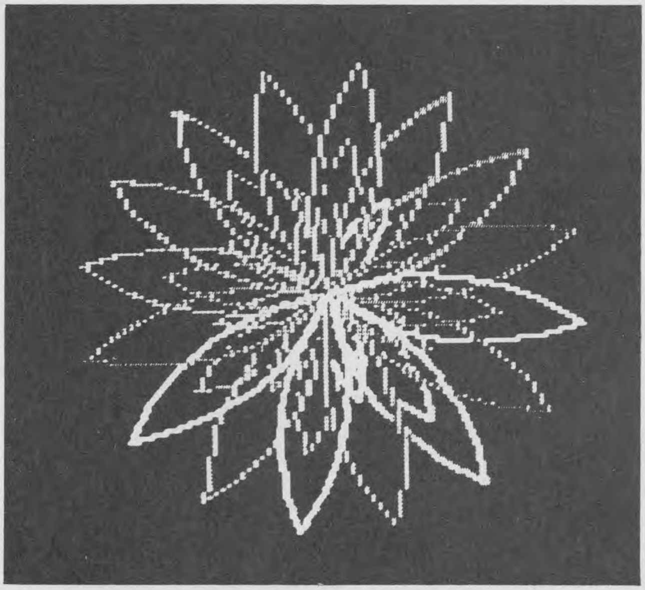
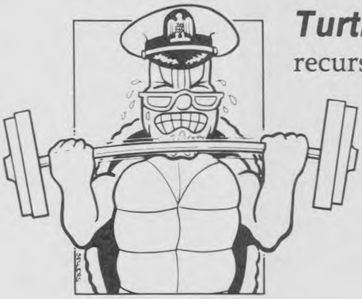


Figure 3-15.

Turtle Tester. Modify the DAISY procedure so that it produces a new flower. Change the number of petals and the proportion of each layer to the others. Save it under a file name of your choice.

Turtle Teaser. Write a series of procedures to draw a row of flowers, with randomly colored petals, across the screen. Save these procedures under a file name of your choice.

Activity 3.5 Turtle Iterations



Turtle Trials. This activity introduces the concept of recursion.

1. Enter the following procedure and describe the result.

```
TO MESSAGEBOX
PR [TURTLE BOX]
END
MESSAGEBOX ________________________________
```

2. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX
PR [TURTLE GRAPHICS]
PR [TURTLE GRAPHICS]
PR [TURTLE GRAPHICS]
END
MESSAGEBOX ________________________________
```

3. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX
CLEARTEXT
REPEAT 3 [PR [TURTLE GRAPHICS]]
END

TEXTCOLOR 1   _____________________________
MESSAGEBOX ________________________________
```

4. Use the editor to modify MESSAGEBOX to read as follows and then describe the results on the lines provided.

```
TO MESSAGEBOX :NUMBER
```

```
CLEARTEXT
REPEAT :NUMBER [PR [TURTLE GRAPHICS]]
END

TEXTCOLOR 7 ______________________________
TEXTBG 4 _________________________________
MESSAGEBOX 10 ____________________________
MESSAGEBOX 1000
CTRL-G (after a while) ___________________
MESSAGEBOX 1000
CTRL-Z (after a while) ___________________
```

NOTE You will now receive the message

```
PAUSE, IN LINE
  REPEAT :NUMBER [PR [TURTLE GRAPHICS]]
AT LEVEL 1 OF MESSAGEBOX.
L1?■
```

At this point, type

```
CO
CTRL-W
```

NOTE At this point printing will stop, but no cursor will appear. Press any key to continue.

```
CTRL-G (after a while) ___________________
DRAW DRAWSTATE ___________________________
```

NOTE A complete description of the information provided by DRAW DRAWSTATE is included in Turtle Talk.

5. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX
PR [TURTLE GRAPHICS]
MESSAGEBOX
END

TEXTCOLOR 0 ______________________________
```

```
TEXTBG 9  ______________________________

MESSAGEBOX
CTRL-G ________________________________
MESSAGEBOX
CTRL-Z ________________________________
CO ____________________________________
CTRL-W ________________________________
(press any key)
CTRL-G ________________________________
```

6. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX :COLORTEXT :COLORBG
TEXTCOLOR :COLORTEXT
TEXTBG :COLORBG
PR [TURTLE GRAPHICS]
MESSAGEBOX :COLORTEXT :COLORBG
END

MESSAGEBOX 1 0  _______________________
CTRL-G
MESSAGEBOX 7 5  _______________________
CTRL-G ________________________________
```

7. Use the editor to modify MESSAGEBOX to read as follows:

```
TO MESSAGEBOX :COLORTEXT :COLORBG
TEXTCOLOR :COLORTEXT
TEXTBG :COLORBG
PR [TURTLE GRAPHICS]
REPEAT 3 [PR "]
MESSAGEBOX :COLORTEXT :COLORBG
END

MESSAGEBOX 0 2
CTRL-W ________________________________
CTRL-Z ________________________________
CTRL-G ________________________________
```

```
CO ________________________________________
CTRL-G ____________________________________
```

8. Enter the following procedure and describe the results.

```
CLEARTEXT
TO DESIGN
ST
REPEAT 4 [FD 40 RT 90]
PR [PROVERBIAL SQUARE]
END

TEXTCOLOR 6
TEXTBG 14
DESIGN ____________________________________
DRAW DRAWSTATE
```

9. Use the editor to modify DESIGN to read as follows, then describe the results.

```
TO DESIGN
ST
REPEAT 4 [FD 40 RT 90]
PR [INFINITE SQUARE]
DESIGN
END

DESIGN
CTRL-G (after a while) __________________
```

Turtle Talk. When you boot the computer with Logo, you are placed in full text mode. The placement of messages depends upon whether full text mode or mixed graphics and text mode is in effect. When in full text screen mode, all messages appear from the top of the screen to the bottom.

In contrast, the mixed graphics and text screen has a five-line text window located at the bottom of the screen. Parts 8 and 9 demonstrate that messages in this mode of operation appear first at the top of the text window, then move toward the bottom. Scrolling (when messages disappear from the text window) occurs when more than five lines of text are printed. The CLEARTEXT command removes all messages from the screen display, regardless of the screen display mode.

The color of the printed messages and the color of the text window background can be changed by using the TEXTCOLOR and TEXTBG commands, respectively. The same sixteen colors are available as for the graphics screen. Parts 3 through 7 demonstrate several possibilities, some more appealing than others. TEXTCOLOR 6 and TEXTBG 14 in Part 9 return the text window to its normal colors when you start the system with Logo.

The DRAWSTATE commands provide nine pieces of information regarding the turtle's current state. In Part 4, DRAW DRAWSTATE replies with the message

```
RESULT: [TRUE TRUE 11 1 DRAW SINGLECOLOR
  SPLITSCREEN 4 7]
```

A description of each state of the turtle is listed below. The possible responses are given within the parentheses.

1. The pen down (TRUE/FALSE)
2. The turtle is visible (TRUE/FALSE)
3. The background color (0, 1, 2, . . . ,15)
4. The pencolor (0, 1, 2, . . . ,15)
5. The immediate mode (DRAW/NODRAW)
6. The color mode (SINGLECOLOR/DOUBLECOLOR)
7. The screen format (SPLITSCREEN/FULLSCREEN/TEXTSCREEN)
8. The text background color (0, 1, 2, . . . , 15)
9. The text color (0, 1, 2, . . . ,15)

Parts 1 through 8 demonstrate a variety of methods by which the repetitive display of a message can appear on the full text screen. Part 5 introduces a technique known as recursion, which is the simplest method of accomplishing a repetitive display.

Recursion allows you to use the procedure name within the procedure so that it executes repeatedly. The difficulty that arises, however, is in stopping the execution of the procedure.

Part 5 also introduces three methods for stopping the execution of a procedure. By pressing CTRL-G, execution of the procedure is terminated and the user is returned to toplevel mode. CTRL-Z behaves as a pause, but when CO (COntinue) is typed, execution of the procedure continues. CTRL-W also stops program execution, but the program proceeds to execute when any key is typed. Activity 3.7 will teach you how to terminate the execution of a procedure that uses recursion by program control, rather than with the CTRL key functions.

Part 6 introduces the use of recursion with procedures that include variables. Notice that all recursive references to procedures must include its variables as well.

Part 9 introduces the use of recursion in graphics mode. While the repetitive drawing of the same square in the same position may not appear to be particularly useful, Activity 3.6 will teach you how to generate some interesting graphic effects by using the principle of recursion.

Turtle Truth. Study the form and content of the following procedures. Type in the commands to verify your deductions. Compare the screen displays to Figure 3-16, Figure 3-17, and Figure 3-18.

```
TO AIR :COLOR
PC :COLOR
BASE
DECK
GUNS
TOWER
END

TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH RT 90 FD :WIDTH RT 90 ]
END

TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO BASE
ST
SETH 270
FD 40
```

```
SETH 135
FD 10
SETH 90
FD 60
SETH 45
FD 10
SETH 270
FD 40
END

TO DECK
BK 40
RECTANGLE 80 5
END

TO GUNS
SETH 0
FD 5
LT 90
FD 5
RECTANGLE 4 2
SETH 0
FD 2
SETH 35
RECTANGLE 6 2
END

TO TOWER
PU
SETH 270
FD 20
PD
RECTANGLE 16 8
SETH 0 FD 8
SETH 270 FD 2
RECTANGLE 12 6
SETH 0 FD 6
SETH 270 FD 2
```

```
RT 90 FD 6
BK 2 SETH 270
PU BK 2 PD
FD 4 PU SETH 180
FD 23 SETH 0
PD HT
END
AIR 7 ____________________________________

TO CONVOY
BG 5
MAKE "COLOR RANDOM 2
REPEAT 3 [AIR :COLOR SETH 0 PU LT 90 FD 280/3 PD]
END

DRAW
CONVOY ___________________________________
```

Edit CONVOY to read as follows:

```
TO CONVOY
BG 6
MAKE "COLOR RANDOM 2
REPEAT 3 [AIR :COLOR SETH 0 PU LT 90 FD 320/3 PD]
PU SETH 0 FD 35 SETH 270 PD
CONVOY
END

DRAW
CONVOY
CTRL-L

CTRL-G (when you are tired of this)
SAVE "SHIPS
GOODBYE
```

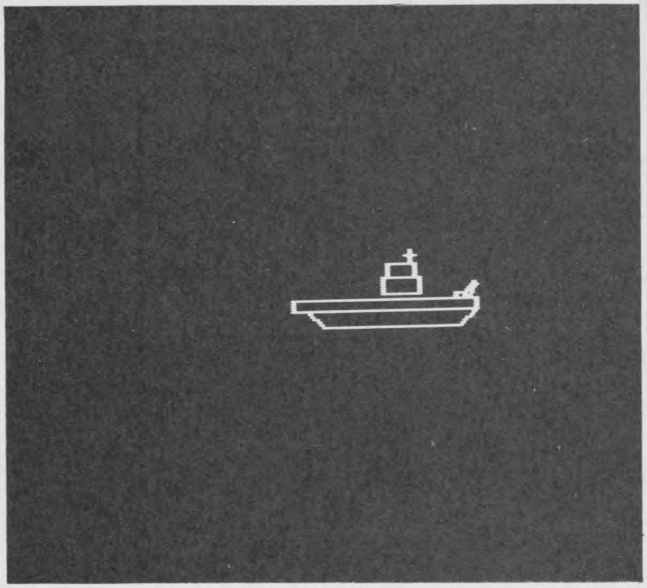


Figure 3-16.

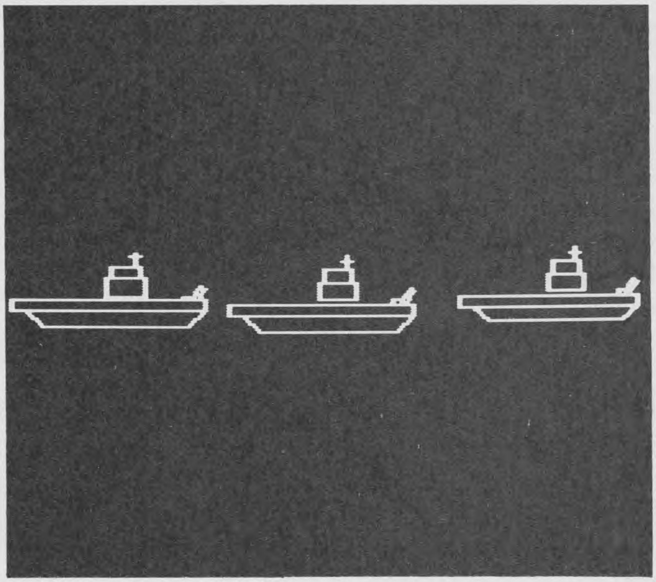


Figure 3-17.

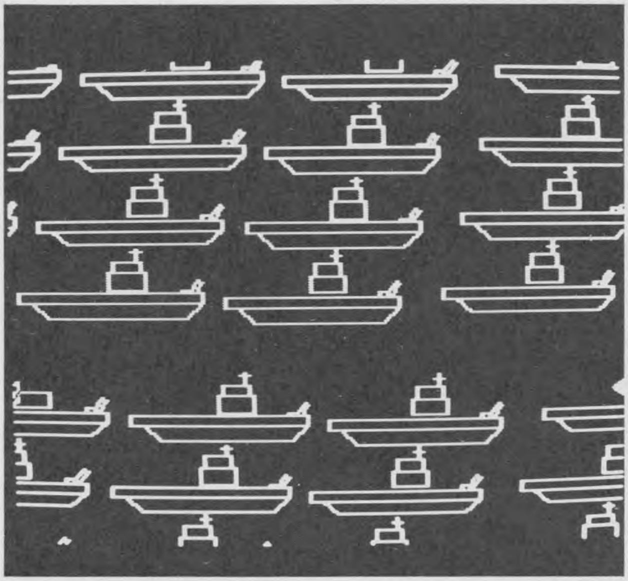


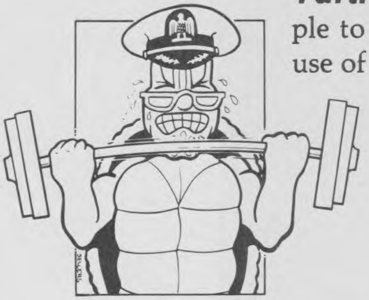
Figure 3-18.

NOTE The MAKE "COLOR command controls the color of each row of carriers. RANDOM 2 randomly selects either color 0 or 1. This insures that the drawing is visible on the blue background.

Turtle Tester. Develop a series of procedures that recursively display an object of your choice across the screen. Include pencolor and background color in conjunction with the MAKE command to generate an interesting display. Save the procedures under a suitable file name.

Turtle Teaser. Use the aircraft carrier or another object and devise a vertical recursive pattern.

Activity 3.6 Turtle Limits



Turtle Trials. This activity uses the recursion principle to draw designs that can be easily varied through the use of input variables.

1. Enter the following procedure and describe the results. (See if you can spot the intentional error.)

```
TO POLY :SIDE
FD :SIDE
RT 90
POLY
END
POLY 40 ___________________________________
```

2. Use the editor to modify POLY to read as follows.

```
TO POLY :SIDE
FD :SIDE
RT 90
POLY :SIDE
END
POLY 40
CTRL-G ____________________________________
DRAW
POLY 60
CTRL-G ____________________________________
```

3. Use the editor to modify POLY as indicated, then describe the results. Compare your screen displays to the illustrations in Figures 3-19, 3-20, and 3-21.

```
TO POLY :SIDE
FD :SIDE
```

```
RT 90
POLY :SIDE + 5
END

POLY 5
CTRL-G (after several seconds) ____________
DRAW
f5 (in full screen mode you will not be
  able to see what you write. Carefully
  enter the next command.)
POLY 10 __________________________________
CTRL-G (after several seconds) ___________
DRAW
NOWRAP
POLY 15
f5
DRAW
f3
WRAP
POLY 12
CTRL-G (after several seconds) ___________
```

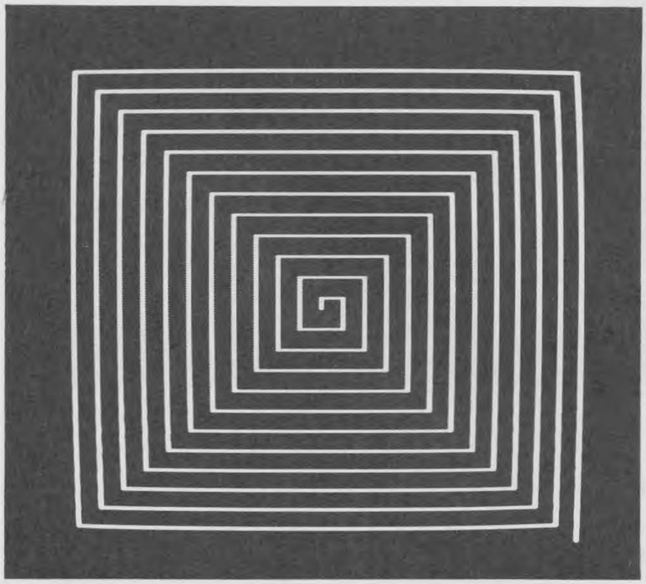


Figure 3-19.

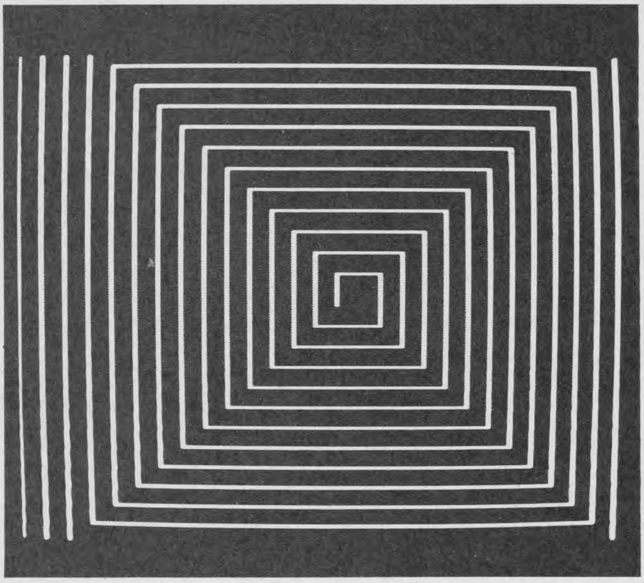


Figure 3-20.

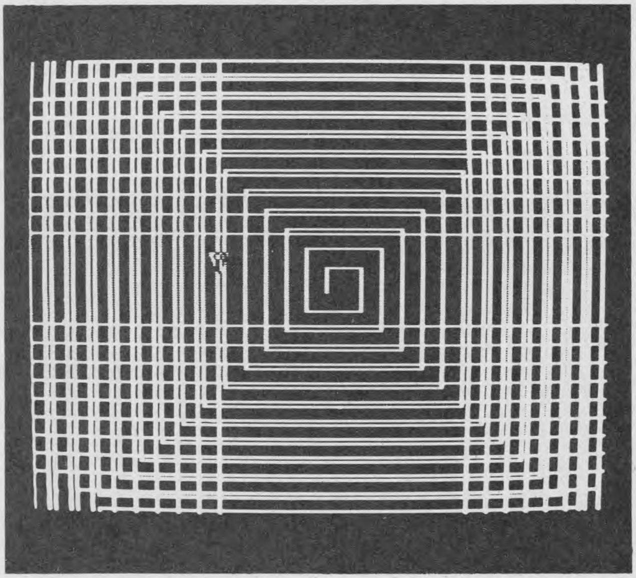


Figure 3-21.

4. Use the editor to modify POLY to read as follows. Then describe the results and compare your second screen display to the illustration in Figure 3-22.

```
TO POLY :SIDE :STEP
```

```
FD :SIDE
RT 90
POLY :SIDE + :STEP :STEP
END

NOWRAP
POLY 10 5 ______________________________
f5
f3
DRAW
POLY 5 1 _______________________________
```

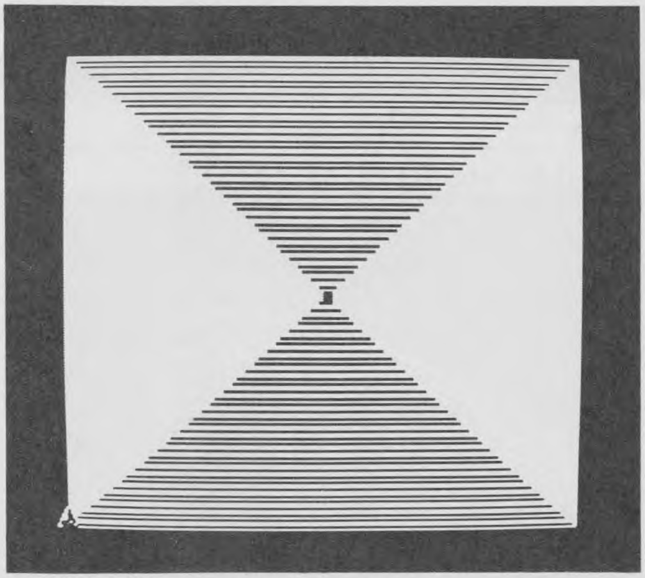


FIGURE 3-22.

5. Enter the new procedure HUE and insert it in POLY as shown below. Then type in the commands that call POLY and observe the interesting effects.

```
TO HUE
MAKE "COLOR RANDOM 11
PC :COLOR
END

TO POLY :SIDE :STEP
```

```
HUE
FD :SIDE
RT 90
POLY :SIDE + :STEP :STEP
END

POLY 10 5  ________________________________
CTRL-G
DRAW
NOWRAP
POLY 3 2  ________________________________
DRAW
POLY 5 1  ________________________________
DRAW
WRAP
POLY 10 3  _______________________________
CTRL-G (when you tire of this)
```

6. Use the editor to modify POLY and enter WAIT to read as follows. Observe the screen displays, and compare them to the diagrams in Figures 3-23 and 3-24. Describe the results on the lines provided.

```
TO POLY :SIDE :STEP :ANGLE
HUE
FD :SIDE
RT :ANGLE
WAIT 3
POLY :SIDE + :STEP :STEP :ANGLE
END

TO WAIT :TIME
REPEAT :TIME*500[]
END

NOWRAP
POLY 2 2 60  ______________________________
DRAW
POLY 5 5 120  _____________________________
```

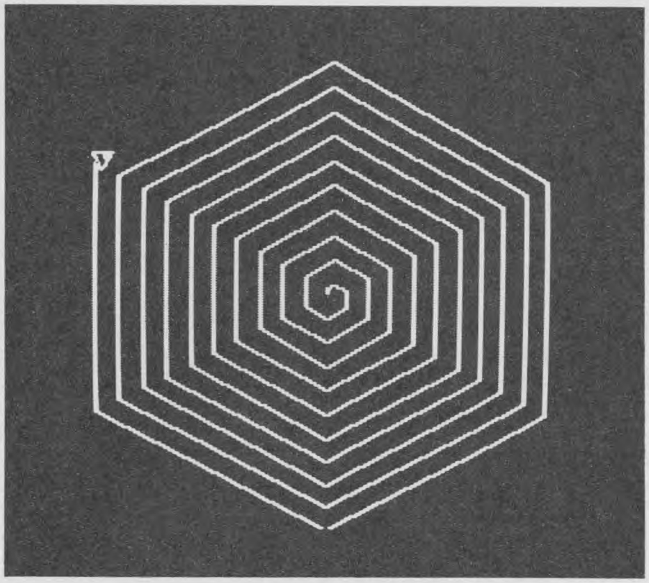


Figure 3-23.

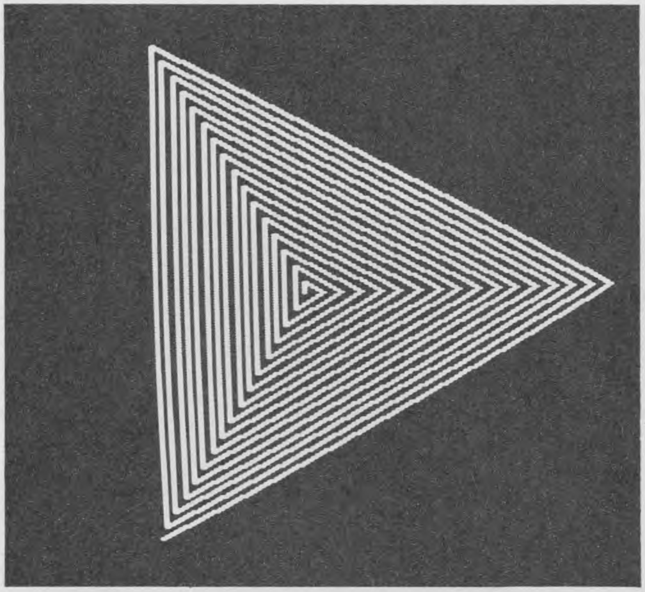


Figure 3-24.

7. Edit POLY so that it is modified to read as follows. Enter the FOLLOW procedure so that you can see the value of the angle change with each step. Upon completion of each procedure, compare your screen displays to Figures 3-25 and 3-26.

```
TO POLY :SIDE :STEP :ANGLE
HUE
FD :SIDE
RT :ANGLE
FOLLOW
POLY :SIDE :STEP :ANGLE + :STEP
END

TO FOLLOW
PR SENTENCE [THE PRESENT ANGLE VALUE IS]
  :ANGLE
WAIT 3
CLEARTEXT
END

DRAW
POLY 20 10 120
CTRL-G (after a while) ____________________
DRAW
POLY 10 5 244
CTRL-G (after a long while) _______________
```

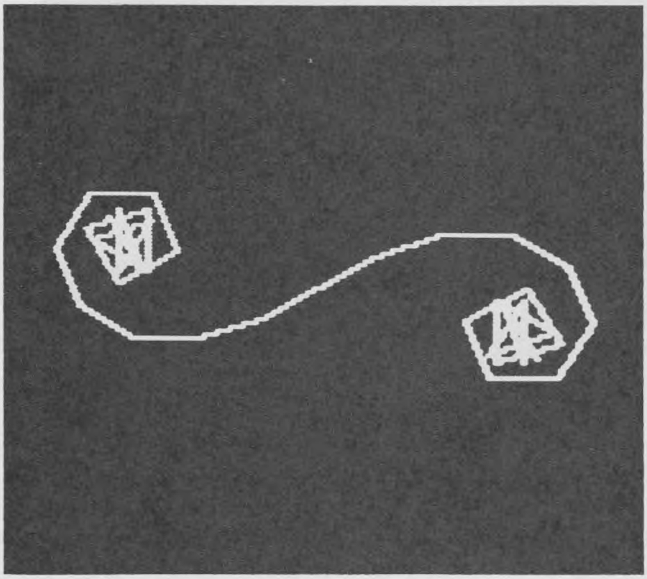


Figure 3-25.

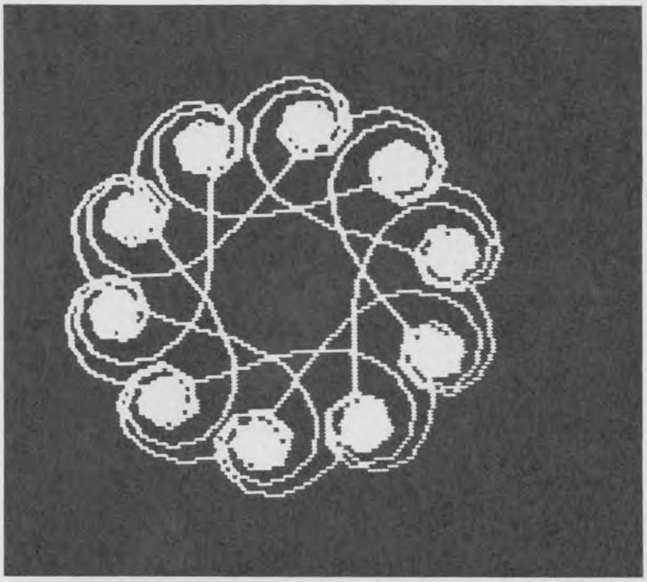


Figure 3-26.

8. Use the editor to modify POLY and FOLLOW to read as follows. Then enter the commands that call POLY and observe the results. Compare your screen displays to the illustrations in Figures 3-27 and 3-28.

```
TO POLY :SIDE :STEP :ANGLE
HUE
FD :SIDE
RT :ANGLE
FOLLOW
POLY :SIDE + :STEP :STEP :ANGLE + :STEP
END

TO FOLLOW
PR SENTENCE [THE CURRENT VALUE FOR SIDE IS] :SIDE
PR SENTENCE [THE CURRENT VALUE FOR ANGLE
  IS] :ANGLE
WAIT 4
END

POLY 5 5 90 ________________________________
f1  ____________________________________
```

NOTE Notice the changes in the values of the variables each time POLY was executed. Observe that SIDE and ANGLE were incremented by five, which is the value of STEP each time. The value of COLOR changed randomly each time.

```
f3
CTRL-G (after several seconds)
DRAW
POLY 1 1 17 ______________________________
f1 ______________________________________
f3 ______________________________________
CTRL-G (after several seconds) __________
```

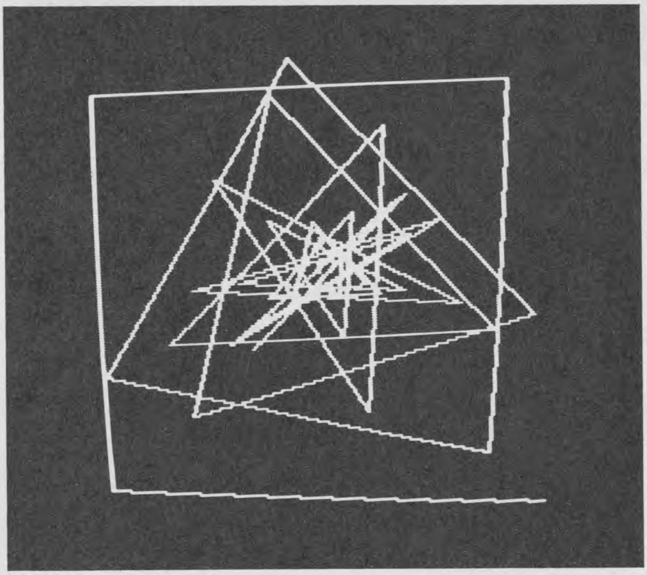


Figure 3-27.

Turtle Talk. Activity 3.5 introduced the concept of recursion with procedures that did not include input variables. This activity expands upon the last by defining recursive-type procedures that do include input variables.

Parts 1 and 2 demonstrate that when a procedure name with input variables is used recursively, the input variables must accompany the procedure name at all times, both when defining the procedure, and when using the procedure name within the procedure.

Parts 3 through 6 generate some interesting designs by changing the val-

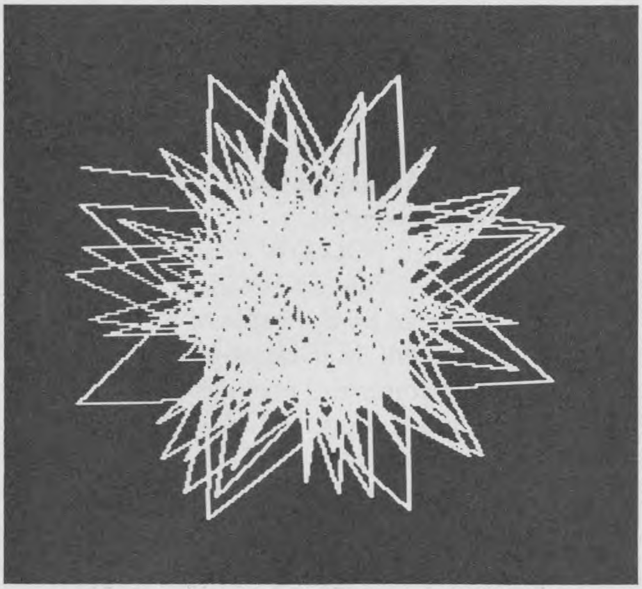


Figure 3-28.

ues of the input variables in the recursive component of the procedures. In Activity 3.5, a square appeared repetitively in the same position by recursively calling the procedure that draws a square. In contrast, Part 3 defines POLY :SIDE, where SIDE is increased by five before each successive execution of POLY giving a spiral-like, box-shaped design, rather than the redrawing of the same object.

However, a new complication appears. The modifying of variables in recursive procedures creates designs that will eventually draw off the screen (as occurs in Part 3) and continue until the user terminates the execution of the program.

The CTRL-G function is a familiar standby for resolving this difficulty. Part 3 introduces another alternative. Preceding the execution of a procedure by the NOWRAP command causes Logo to give a TURTLE OUT OF BOUNDS error when the drawing occurs outside the screen boundaries. The WRAP command puts Logo into its default mode, where drawings that appear outside the screen boundaries continue to draw (wrap around) in the indicated path. The WAIT procedure used in Part 6 slows the speed at which the turtle draws a design.

Part 5 adds color to the procedure POLY. It is added by defining a procedure named Hue and including it as the first line in POLY. HUE randomly generates a pencolor to be used in POLY each time POLY executes. The color gray (or 11) was eliminated through the use of RANDOM 11 in the procedure (the manner in which this was done was explained in Activity

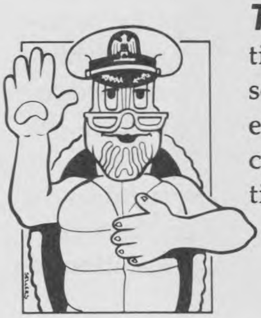
3.4). Color not only adds interest, but lets the viewer easily distinguish each side that is drawn.

Parts 4 through 7 redefine POLY in terms of more than one input variable. A modification of these variables in recursively defined procedures creates some fascinating and, perhaps, unexpected designs. It is interesting to experiment by supplying different input values for these variables to observe what other displays are possible.

In Part 7, a procedure named FOLLOW is added to POLY. It executes after each POLY and prints the value of ANGLE. This lets the viewer watch ANGLE grow as it is incremented by the value of STEP. Examine the structure of the print (PR) statement that displays the value of angle. The SENTENCE (abbreviated SE) command provides for the printing of a message and the value of a variable on one line. Also notice ANGLE is defined in FOLLOW since FOLLOW is included in POLY.

Part 8 expands FOLLOW so the values for two variables are printed after each execution of POLY. Because of the limited text window, you can not read the text easily while the procedure is running. The use of f1 after the procedure stops lets the viewer study the display produced by FOLLOW.

Activity 3.7 will show you how to control the graphics screen display with program control, rather than by using NOWRAP or CTRL-G.



Turtle Truth. Read the ARCS file from the Logo Utilities Disk. Study the procedures below and imagine the screen display. Type in the procedures and examine the effects of recursion and the use of incrementation. Upon completion, compare your screen displays to the illustrations in Figures 3-29 and 3-30.

```
TO OVAL :LENGTH :WIDTH
FD :LENGTH/2
ARCR :WIDTH/2 180
FD :LENGTH
ARCR :WIDTH/2 180
FD :LENGTH/2
END

OVAL 50 10

TO OVALS :LENGTH :WIDTH
REPEAT 6 [OVAL :LENGTH :WIDTH RT 60]
```

```
END

OVALS 50 10
TO MESSAGE
PR SE [THE OVAL'S LENGTH IS] :LENGTH
PR SE [THE OVAL'S WIDTH IS] :WIDTH
WAIT 3
CLEARTEXT
END

TO DESIGN :LENGTH :WIDTH :STEP
HUE
OVALS :LENGTH :WIDTH
MESSAGE
DESIGN :LENGTH + :STEP :WIDTH + :STEP
  :STEP
END

NOWRAP
f5
DESIGN 50 10 40
DRAW
DESIGN 80 30 20
DRAW
```

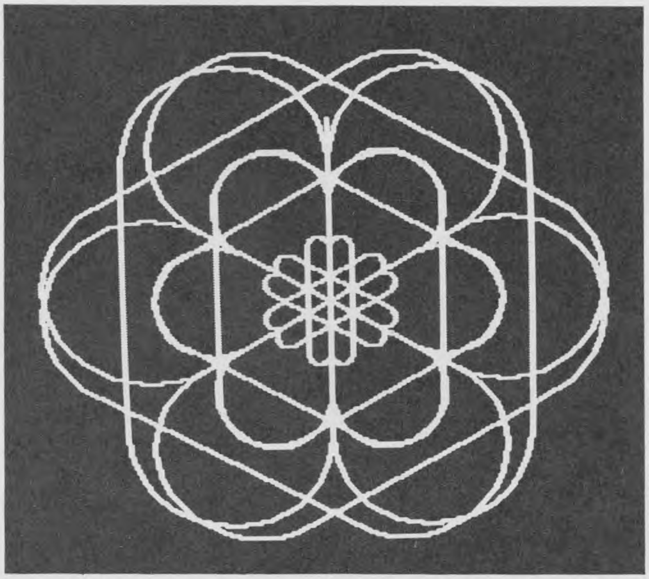


Figure 3-29.

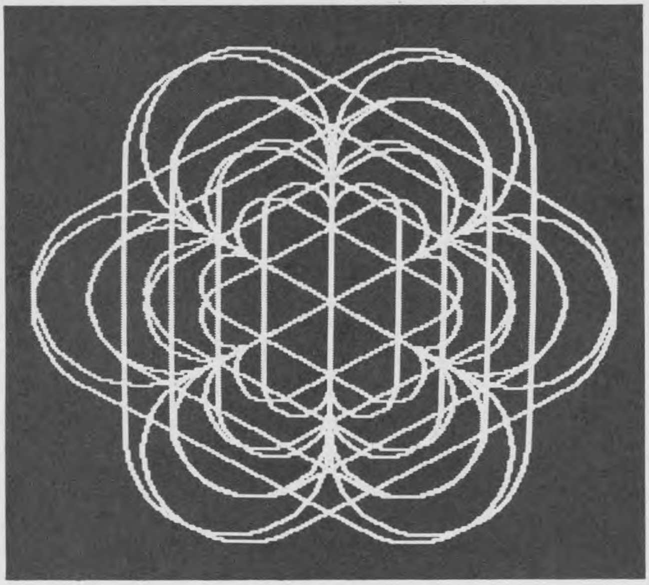


Figure 3-30.

NOTE Save these procedures in a file named RECURSION.PICS. Add the following Turtle Tester and Turtle Teaser procedures to the file RECURSION.PICS or save it separately under another name.

Turtle Tester. Write a CIRCLES procedure that creates the diagram in Figure 3-31. Then create a recursive procedure named MANY.CIRCLES that draws CIRCLES with an initial radius of 30 and an incrementation of 10 until NOWRAP stops the procedure. Your first procedure should have a display similar to Figure 3-31 and the recursive procedure should be similar to Figure 3-32. (Answer on page 290.)

Turtle Teaser. Add color and a message to the MANY.CIRCLES procedure. The message should print the current radius of the circle and include a WAIT procedure permitting the user time to read the output.

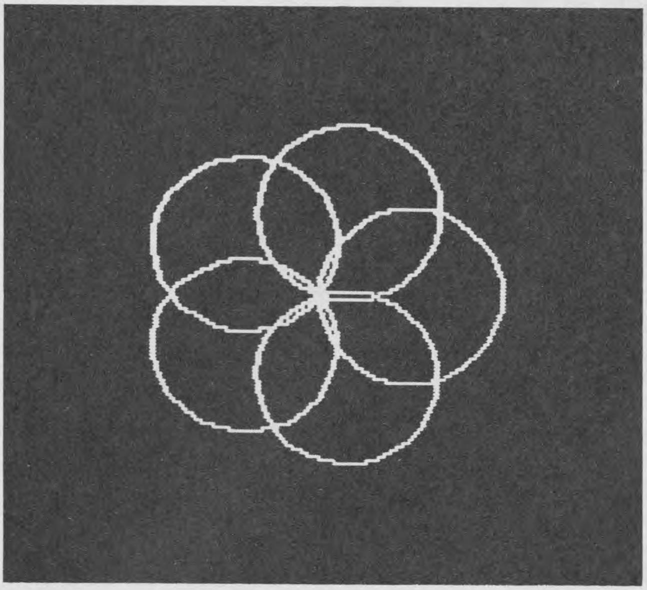


Figure 3-31.

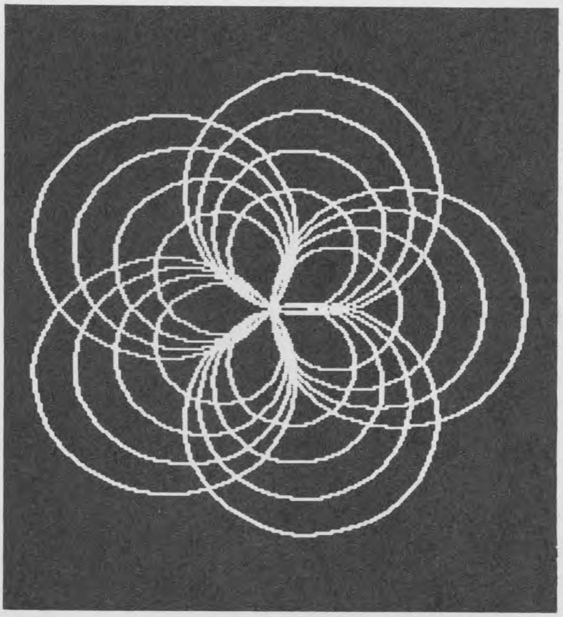
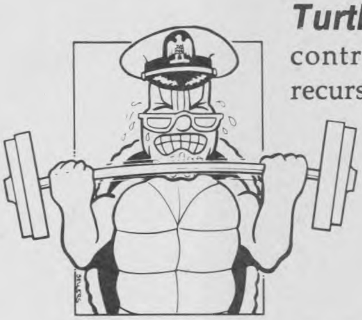


Figure 3-32.

Activity 3.7 Turtle Conditions



Turtle Trials. This activity shows you how to gain control of procedures that include simple uses of recursion.

1. Enter the following procedure and describe the result.

```
TO TALLY :NUMBER
TEXTSCREEN
PR :NUMBER
TALLY :NUMBER + 1
END

TALLY 1
CTRL-G (after 10 seconds) ________________
```

2. Modify TALLY to read as follows and describe the result.

```
TO TALLY :NUMBER
TEXTSCREEN
IF :NUMBER = 21 STOP
PR :NUMBER
TALLY :NUMBER + 1
END
TALLY 1 ________________________________
```

NOTE The symbols = (equal), > (greater than), and < (less than) are used in predicates. Predicates are expressions that return the value TRUE or FALSE. For example, 6 > 3 is a predicate with value TRUE and 7 < 4 is a predicate with value FALSE.

3. Enter the following commands and record your observations below.

```
CLEARTEXT
TO MANYBOX :SIDE
```

```
REPEAT 4 [FD :SIDE RT 90]
MANYBOX :SIDE + 2
END

MANYBOX 10
CTRL-G (after 15 seconds) ____________________
```

4. Modify MANYBOX to read as indicated below and describe the results. Compare your second screen display to the illustration in Figure 3-33.

```
TO MANYBOX :SIDE
IF :SIDE > 100 STOP
REPEAT 4 [FD :SIDE RT 90]
MANYBOX :SIDE + 2
END

MANYBOX 10 ________________________________________
DRAW
MANYBOX -100 _____________________________________
f5
DRAW
f3
```

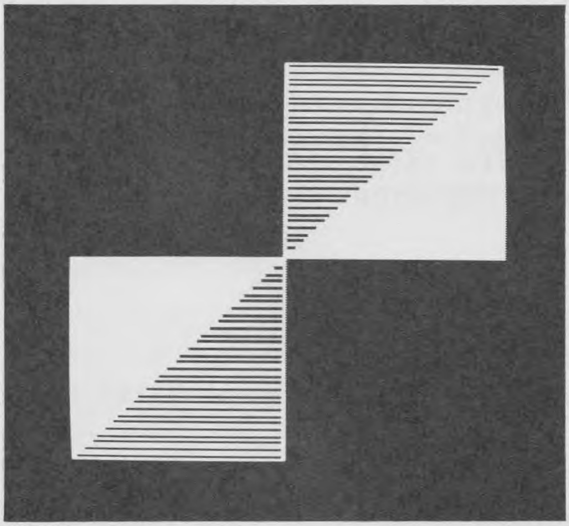


Figure 3-33.

5. Add a WAIT procedure and use the editor to modify the MANYBOX procedure to add color and a print statement. The new procedure should read as follows:

```
TO MANYBOX2 :SIDE :COLOR
IF :COLOR > 10 MAKE "COLOR 0
PC :COLOR
IF :SIDE > 70 STOP
REPEAT 4 [FD :SIDE RT 90]
PR SE [THE PENCOLOR IS] :COLOR
PR SE [THE SIDE LENGTH IS] :SIDE
WAIT 2
MANYBOX2 :SIDE + 5 :COLOR + 1
END

TO WAIT :TIME
REPEAT :TIME*500 []
END

MANYBOX2 -70 0  ____________________________
```

NOTE COLOR becomes a local variable when MANYBOX2 is defined. The IF :COLOR > 10 MAKE "COLOR 0 command in MANYBOX2 does not cause COLOR to become a global variable. As an exception to the definition given in Activity 3.4, MAKE defines a global variable only if it has not been previously defined as a local variable.

6. Enter the following procedure and record the results.

```
TO POLY :SIDE :STEP :ANGLE
IF :SIDE > 100 STOP
MAKECOLOR
PC :COLOR
FD :SIDE
RT :ANGLE
POLY :SIDE + :STEP :STEP :ANGLE
END

TO MAKECOLOR
MAKE "COLOR RANDOM 16
```

```
IF :COLOR = 11 MAKECOLOR
END

POLY 2 4 72 ______________________________
DRAW
POLY 10 5 120 ____________________________
DRAW
POLY 5 4 245 _____________________________
DRAW
POLY 8 1 315 _____________________________
```

7. Modify POLY to read as follows and describe the results.

```
TO POLY :SIDE :STEP :ANGLE :NUMBER
IF :NUMBER = 0 STOP
FD :SIDE
RT :ANGLE
POLY :SIDE :STEP :ANGLE + :STEP
  :NUMBER - 1
END

POLY 5 5 244 10 __________________________
DRAW
POLY 5 5 244 200 _________________________
DRAW
POLY 5 5 244 1000 ________________________
```

8. Edit POLY to read as follows and describe the resulting screen displays.

```
TO POLY :SIDE :STEP :ANGLE :NUMBER
IF :NUMBER < 0 STOP
FD :SIDE
RT :ANGLE
PR :NUMBER
POLY :SIDE :STEP :ANGLE + :STEP
  :NUMBER - 1
END

POLY 5 5 244 1000 ________________________
```

```
DRAW
POLY 5 5 244 670 ____________________________
DRAW
POLY 4 4 180 120 ____________________________
DRAW

PU RT 90 FD 138 LT 90 PD
POLY 4 4 330 620 ____________________________
```

Turtle Talk. The IF statement is a conditional expression. It provides a test to determine whether a condition is true or false. If the condition tested is true, the program execution is passed to the list enclosed in the brackets following the expression. If the tested condition is false, the program execution passes to the next line in the procedure.

When using recursion, the IF statement provides a means of programming the termination of a procedure. Part 1 presents a recursive procedure that displays the counting numbers beginning with one and continuing forever, if permitted. Part 2 adds the IF command to TALLY achieving program control.

TALLY 1 begins execution of the program and displays the number 1 first. The statement TALLY :NUMBER + 1, within the procedure TALLY :NUMBER, recursively calls upon TALLY :NUMBER, but with an input value of one more than the previous call. This recursion has the effect of performing successively TALLY 1, TALLY 2, TALLY 3, . . . TALLY 20, if the procedure TALLY is written as follows:

```
TO TALLY :NUMBER
IF :NUMBER = 21 STOP
PR :NUMBER
TALLY :NUMBER + 1
END
```

By using recursion, the desired outcome is performed more simply and efficiently. In TALLY, the value assigned to NUMBER increases by 1 with each repetition. When NUMBER reaches a value of 21, the program terminates because of the STOP command following the IF statement. Until then, the condition in the IF statement is false; thus the flow of the program proceeds to the next statement, PR :NUMBER, where the value of NUMBER is displayed.

The STOP command, because of its denotation, does not have the effect on the program that you might expect. It does not "stop" the program. Instead, it returns control to wherever the procedure originally is called. Since TALLY is entered from toplevel mode, the STOP command returns the program to toplevel, giving the appearance of stopping the program. But if TALLY has been called upon by another procedure, program control would be returned to that procedure. Activity 3.8 will expand your understanding of this concept when more complex situations using recursion are presented.

Parts 4 through 8 demonstrate the use of the IF statement to control recursive procedures that involve graphics. Three types of conditions that can be tested involve expressions called predicates. Part 2 demonstrates = (equal), which tests equality of two numbers. Part 4 uses > (greater than), which tests whether the number on the left is greater than the number on the right. Similarly, < (less than) in Part 8 is demonstrated.

Part 6 also introduces a procedure to assure that the pencolor is not identical to the background color. MANYCOLOR uses recursion to randomly select a color number from 0 to 15, inclusively. If 11 (the background color) is selected, MANYCOLOR calls upon itself to generate another number until 11 is not generated. When a number other than 11 is obtained, the POLY procedure continues. Notice MANYCOLOR defines COLOR to be a global variable and, thus, it need not be defined in POLY.

In parts 7 and 8, one becomes curious as to what designs can be created by experimenting or "playing" with input values by modifying variables used in the graphic recursion procedures.

In Part 7, POLY 5 5 244 10 is not particularly exciting. POLY 5 5 244 200 generates a design of more interest, but it is not complete. POLY 5 5 244 1000 not only completes the design, but it also begins to redraw it.

The PR :NUMBER command, inserted in the editing of POLY in Part 8, provides a means by which the proper value for NUMBER in POLY can be determined. POLY 5 5 244 1000 now displays a value of approximately 330 when the diagram is completed, suggesting that 670 (1000 - 330) is an appropriate value for NUMBER to obtain the completed diagram.

Other input values for POLY in Part 8 generate designs that might be of interest to the reader.

Turtle Truth. Often the creation of a Logo procedure is actually a multi-step process. The original idea is accomplished through the building of an initial procedure, but upon reflection, a certain series of commands appears in several places. This pattern suggests that an additional procedure could replace the commands and produce a more concise product. As one studies the manner in which variables are defined, it may suggest that the use of global variables is more effective than local variables. Study the following procedures as they evolve into a final form. Identify where procedures are obtaining values for variables. Compare your screen displays to the illustrations in the figures below each section.

```
TO TRI :SIDE :STEP
IF :SIDE < 3 STOP
REPEAT 3 [FD :SIDE LT 120]
TRI :SIDE - :STEP :STEP
END

TRI 100 3 ____________________________________
DRAW
TRI 100 5 ____________________________________
TO START :SIDE :STEP
PC 8
RT 90
TRI :SIDE :STEP
END

START 120 5 __________________________________

TO FILL :SIDE :STEP :MANY
IF :SIDE < :STEP STOP
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :STEP :STEP :MANY
END

FILL 100 3 3 _________________________________
DRAW
FILL 100 1 3 _________________________________
```

```
DRAW
FILL 60 .5 6 ________________________________
```

NOTE The FILL procedure shades or "fills" polygons with different degrees of density depending upon the value for STEP. Figures 3-34, 3-35 and 3-36 illustrate some possible effects.

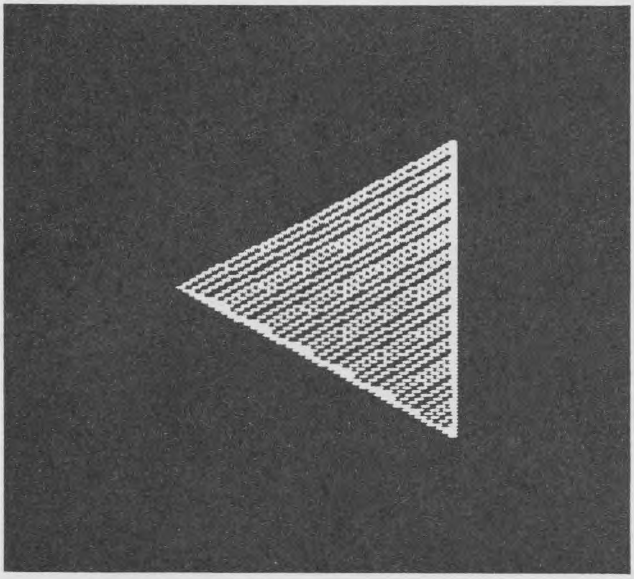


Figure 3-34.

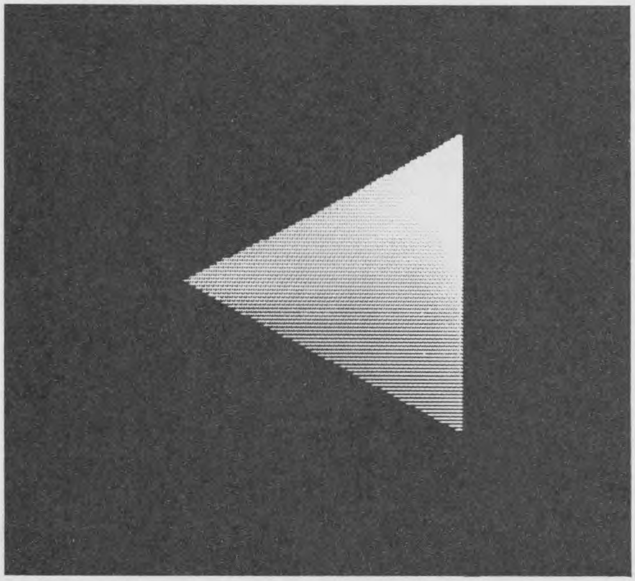


Figure 3-35.

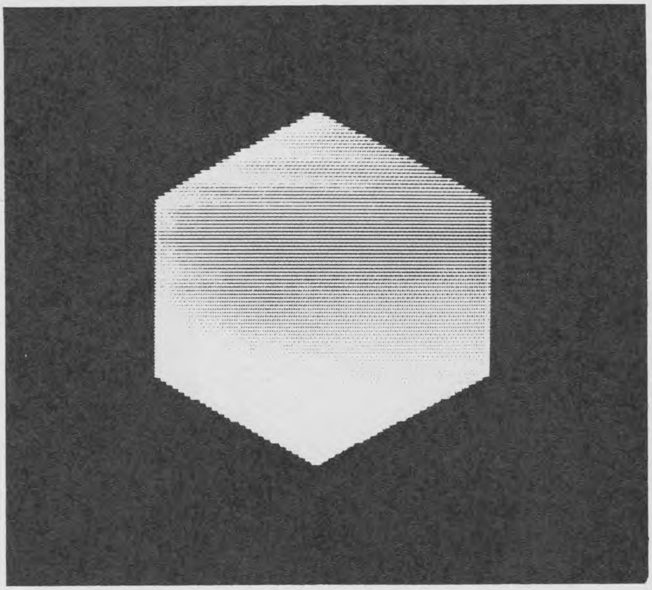


Figure 3-36.

```
TO COLORFILL :SIDE :STEP :MANY :COLOR
PC :COLOR
FILL :SIDE :STEP :MANY
END

COLORFILL 30 2 8 4 ________________________
DRAW
COLORFILL 50 5 7 2 ________________________

TO HOUSE :SIDE :COLORHOUSE :COLORROOF
PC :COLORHOUSE
MAKE "MANY 4
MAKE "COLOR :COLORHOUSE
MAKE "STEP 1
RT 90
COLORFILL :SIDE :STEP :MANY :COLOR
LT 90
FD :SIDE
PC :COLORROOF
```

```
MAKE "MANY 3
MAKE "COLOR :COLORROOF
MAKE "STEP 3
RT 90
COLORFILL :SIDE :STEP :MANY :COLOR
PU LT 90 BK :SIDE PD
END

HOUSE 50 5 7
LT 90 PU FD 85 RT 90 PD

HOUSE 65 6 1

TO MOVE :ANGLE :MORE
PU SETH :ANGLE FD :MORE SETH 0 PD
END

MOVE 90 165

HOUSE 25 15 10

MOVE 270 225

HOUSE 35 2 5

TO HOUSES
HOUSE 50 5 2
MOVE 270 85
HOUSE 65 3 4
MOVE 90 155
HOUSE 25 4 1
MOVE 270 208
HOUSE 35 2 5
PU HOME PD
HT
END

HOUSES
```

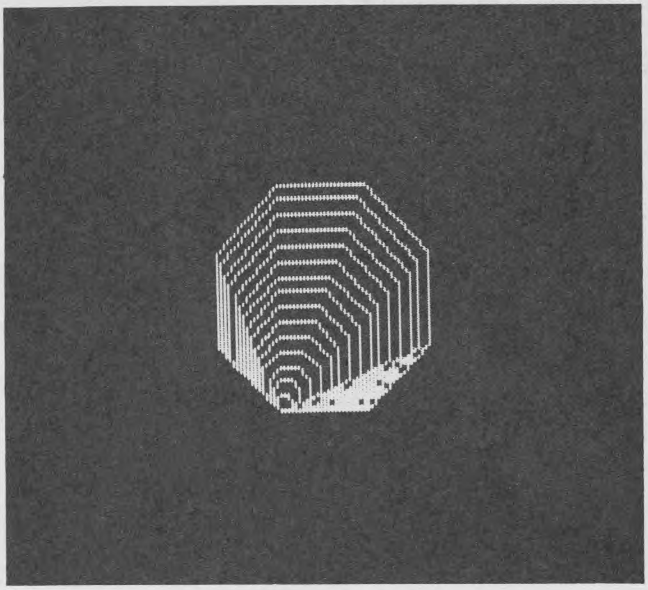


Figure 3-37.

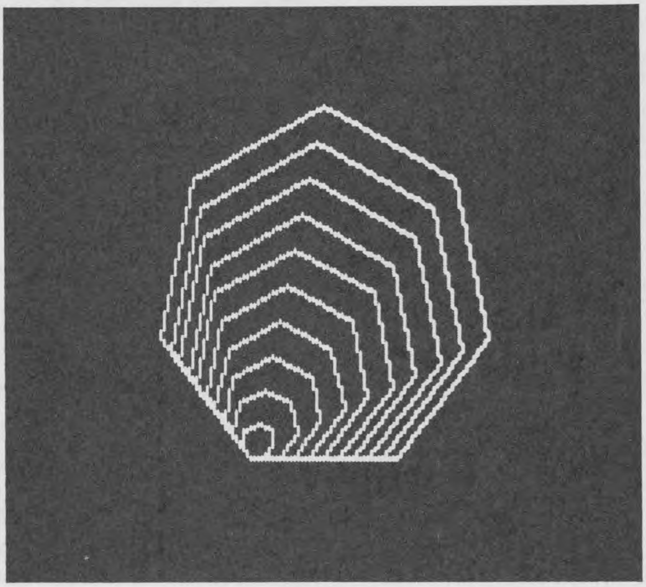


Figure 3-38.

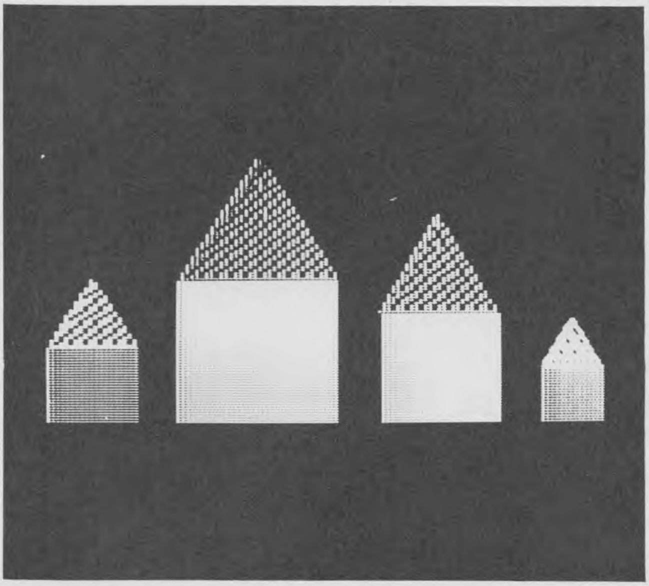


Figure 3-39.

NOTE The MAKE command in procedure HOUSE declares NUMBER, COLOR, and STEP to be global variables. Therefore, these input variables need not be included in the procedure headings for FILL and COLORFILL. Edit FILL, COLORFILL, and HOUSE to read as indicated below. Enter HOUSES to determine if the same screen display appears.

```
TO FILL :SIDE
IF :SIDE < :STEP STOP
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :STEP
END

TO COLORFILL :SIDE
PC :COLOR
FILL :SIDE
END

TO HOUSE :SIDE :COLORHOUSE :COLORROOF
PC :COLORHOUSE
MAKE "MANY 4
MAKE "COLOR :COLORHOUSE
MAKE "STEP 1
RT 90
COLORFILL :SIDE
LT 90
FD :SIDE
PC :COLORROOF
MAKE "MANY 3
MAKE "COLOR :COLORROOF
MAKE "STEP 3
RT 90
COLORFILL :SIDE
PU LT 90 BK :SIDE PD
END
```

HOUSES ____

NOTE You might have observed that there is another improvement to be made in HOUSE. MANY, COLOR, and STEP are defined twice in HOUSE. A better Logo structure is to define a procedure to perform this task. Modify HOUSE and enter GLOBALVALUES to read as follows. Enter HOUSES to verify that the same screen display occurs.

```
TO GLOBALVALUES :FIRST :SECOND :THIRD
MAKE "MANY :FIRST
MAKE "COLOR :SECOND
MAKE "STEP :THIRD
END

TO HOUSE :SIDE :COLORHOUSE :COLORROOF
PC :COLORHOUSE
GLOBALVALUES 4 :COLORHOUSE 1
RT 90
COLORFILL :SIDE
LT 90
FD :SIDE
PC :COLORROOF
GLOBALVALUES 3 :COLORROOF 3
RT 90
COLORFILL :SIDE
PU LT 90 BK :SIDE PD
END
```

```
HOUSES
```

NOTE Notice the "bleeding" affect of colors where the roof of the house meets its upper frame. When colored lines are drawn too close in SINGLECOLOR mode, this problem can occur. Modify HOUSES to include DOUBLECOLOR as follows. Enter HOUSES to observe the improved clarity in the picture.

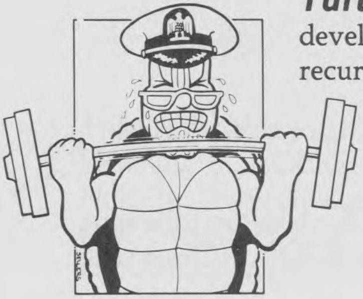
```
TO HOUSES
DOUBLECOLOR
HOUSE 50 5 2
MOVE 270 8 5
HOUSE 65 3 4
MOVE 90 155
HOUSE 25 4 1
MOVE 270 208
HOUSE 35 2 5
PU HOME PD
END

HOUSES
```

Save these procedures in a file named SOLID.PICS. Add the following Turtle Tester and Turtle Teaser procedures to the file SOLID.PICS or save them separately under another name.

Turtle Tester. Use COLORFILL and the MOVE procedures to position shaded colored shapes in the four corners of the screen. Put a purple pentagon in the upper left corner, an orange octagon in the upper right corner, a blue hexagon in the lower right corner, and a green decagon in the lower left corner. (Answer on page 290.)

Turtle Teaser. Draw a robot with each section in a different color. Incorporate the COLORFILL procedure.

Activity 3.8 Turtle Tricks



Turtle Trials. This activity demonstrates how to develop procedures that include more complex uses of recursion.

1. Enter the procedure and describe the result.

```
TO TALLY :NUMBER
IF :NUMBER = 10 STOP
PR :NUMBER
TALLY :NUMBER + 1
END

TALLY 1 ______________________________
```

2. Modify TALLY to read as follows and record the result.

```
TO TALLY :NUMBER
IF :NUMBER = 10 STOP
TALLY :NUMBER + 1
PR :NUMBER
END
```

```
TALLY 1 ______________________________
TRACE
TALLY 1 ______________________________
```

(Each time the program execution stops, press any key to continue).

3. Type the following procedures as indicated and describe the results.

```
NOTRACE
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO CHANGESQUARE :SIDE
IF :SIDE < 4 STOP
SQUARE :SIDE
CHANGESQUARE :SIDE*.5
END

CHANGESQUARE 120 ______________________
```

4. Modify CHANGESQUARE to read as follows and describe the result.

```
TO CHANGESQUARE :SIDE
IF :SIDE < 4 STOP
CHANGESQUARE :SIDE*.5
SQUARE :SIDE
END

CHANGESQUARE 120 ______________________
CS
```

5. Enter the following procedure and describe the result.

```
TO GYROSQUARE :SIDE
IF :SIDE < 8 STOP
SQUARE :SIDE
FD :SIDE/2 RT 45
GYROSQUARE :SIDE*.707
HT
```

```
END

GYROSQUARE 120 ______________________________
```

6. Modify GYROSQUARE to read as follows and record the result.

```
TO GYROSQUARE :SIDE
IF :SIDE < 8 STOP
GYROSQUARE :SIDE*.707
SQUARE :SIDE
LT 45 BK :SIDE*.707
HT
END

GYROSQUARE 120 ______________________________
```

7. Edit GYROSQUARE to read as follows and record your observations.

```
TO GYROSQUARE :SIDE :NUMBER
IF :SIDE < 8 STOP
GYROSQUARE :SIDE*.707 :NUMBER
SQUARE :SIDE
IF :SIDE = :NUMBER STOP
LT 45 BK :SIDE*.707
HT
END

GYROSQUARE 120 ______________________________
GYROSQUARE 120 120 __________________________
```

8. Add MAKECOLOR and WAIT, then edit GYROSQUARE to add pencolor and a delay to the procedure. The change in pencolor before each execution of the procedure assists you in observing the sequence of commands that produce each square. MAKECOLOR prevents a pencolor of black (0) from being generated, since the background is changed to black.

```
TO GYROSQUARE :SIDE :NUMBER
IF :SIDE < 8 STOP
GYROSQUARE :SIDE*.707 :NUMBER
MAKECOLOR
```

```
PC :COLOR
WAIT 2
SQUARE :SIDE
IF :SIDE = :NUMBER STOP
LT 45 BK :SIDE*.707
END

TO MAKECOLOR
BG 0
MAKE "COLOR (1 + RANDOM 14)
IF :COLOR = 0 MAKECOLOR
END

TO WAIT :TIME
REPEAT :TIME*500[]
END

GYROSQUARE 150 150  ______________________
DRAW
GYROSQURE 175 175  ________________________
CTRL-L
DRAW
```

Turtle Talk. In Activities 3.6 and 3.7, the concept of simple recursion was introduced. Part 1 of this activity provides another example of simple recursion, while the example in Part 2 introduces complex recursion. Table 3-2 illustrates the output generated by each of these parts.

With simple recursion, as in Part 1, each recursive call is preceded by the printing of value NUMBER in TALLY. For example, TALLY 1 first prints a "1" before calling TALLY 2, which prints a "2." Similarly, the numbers three through nine are printed.

When the STOP command is executed (for TALLY 10), program execution is transferred to the command following the last recursion, namely the END command, which terminates the program.

Table 3-2. Output generated by Parts 1 and 2.

Part 1 (Simple Recursion)	Part 2 (Complex Recursion)
TALLY 1 - - - - - 1	TALLY 1 - - - - - TALLY 2 (no output)
TALLY 2 - - - - - 2	TALLY 2 - - - - - TALLY 3 (no output)
TALLY 3 - - - - - 3	TALLY 3 - - - - - TALLY 4 (no output)
TALLY 4 - - - - - 4	TALLY 4 - - - - - TALLY 5 (no output)
TALLY 5 - - - - - 5	TALLY 5 - - - - - TALLY 6 (no output)
TALLY 6 - - - - - 6	TALLY 6 - - - - - TALLY 7 (no output)
TALLY 7 - - - - - 7	TALLY 7 - - - - - TALLY 8 (no output)
TALLY 8 - - - - - 8	TALLY 8 - - - - - TALLY 9 (no output)
TALLY 9 - - - - - 9	TALLY 9 - - - - - TALLY 10(calls TALLY 9 because of STOP
TALLY 10 - - - - ?	TALLY 9 - - - - - 9
(program terminates)	TALLY 8 - - - - - 8
	TALLY 7 - - - - - 7
	TALLY 6 - - - - - 6
	TALLY 5 - - - - - 5
	TALLY 4 - - - - - 4
	TALLY 3 - - - - - 3
	TALLY 2 - - - - - 2
	TALLY 1 - - - - - 1
	?
	(program terminates)

In contrast to Part 1, since the PR command in Part 2 follows the recursive formula TALLY :NUMBER + 1, the various recursion calls accumulate and define separate libraries or memory locations associated with the different values for NUMBER. Notice that no printing occurs until after the STOP command executes for TALLY 10. At this point, program execution is transferred to the most previously called recursion (TALLY 9) and proceeds to execute the commands (in this case, PR :NUMBER) following TALLY :NUMBER + 1. This process continues until TALLY 1 is performed. Then the program stops.

The TRACE command lets you observe which command is executing by stopping the program. Pressing any key continues the program. You should have noticed a sequence of messages EXECUTING TALLY 1, EXECUTING TALLY 2, . . . to EXECUTING TALLY 10, followed by ENDING TALLY 9, ENDING TALLY 8, to ENDING TALLY 1. By using TRACE, the principle of complex recursion is verified. NOTRACE returns the system to its normal mode of execution.

Parts 3 and 4 illustrate the differences between simple and complex recursions for a situation that involves graphics. Table 3-3 traces the program flow for each of these parts.

Table 3-3. Program flow for Parts 3 and 4.

PART 3 (SIMPLE RECURSION)

CHANGESQUARE 120 - - - - Draws SQUARE with side = 120
CHANGESQUARE 60 - - - - Draws SQUARE with side = 60
CHANGESQUARE 30 - - - - Draws SQUARE with side = 30
CHANGESQUARE 15 - - - - Draws SQUARE with side = 15
CHANGESQUARE 7 - - - - Draws SQUARE with side = 7
CHANGESQUARE 3 - - - - Draws SQUARE with side = ?
(program terminates)

PART 4 (COMPLEX RECURSION)

CHANGESQUARE 120 - - - - CHANGESQUARE 60 (no output)
CHANGESQUARE 60 - - - - CHANGESQUARE 30 (no output)
CHANGESQUARE 30 - - - - CHANGESQUARE 15 (no output)
CHANGESQUARE 15 - - - - CHANGESQUARE 7 (no output)
CHANGESQUARE 7 - - - - CHANGESQUARE 3 (calls CHANGESQUARE 7 because of STOP)
CHANGESQUARE 7 - - - - Draws SQUARE with side = 7
CHANGESQUARE 15 - - - - Draws SQUARE with side = 15
CHANGESQUARE 30 - - - - Draws SQUARE with side = 30
CHANGESQUARE 60 - - - - Draws SQUARE with side = 60
CHANGESQUARE 120 - - - - Draws SQUARE with side = 120
?(program terminates)

Parts 5 and 6 contrast the two types of recursion, but with a more complex graphics design. The reader with a moderate mathematical background may wish to justify the existence of the various commands and values used to create the design. Otherwise, simply enjoy the effect.

Turtle Truth. Study the following Logo procedures to determine the screen display. Check your answer by entering the commands.

```
TO TREER :RIGHT
IF :RIGHT < 10 STOP
TREER :RIGHT – 15
SETH 120
FD :RIGHT + 5
SETH 270
FD :RIGHT
END

TO TREEL :LEFT
IF :LEFT < 10 STOP
SETH 270
FD :LEFT
SETH 60
FD :LEFT + 5
TREEL :LEFT – 15
END

TO FILL :SIDE :STEP :MANY
IF :SIDE < :STEP STOP
REPEAT :MANY [ XD :SIDE LT 360/:MANY]
FILL :SIDE - :STEP :STEP :MANY
END

TO TREE :LEFT :RIGHT :SIDE :STEP :MANY
BG 1
PC 5
TREEL :LEFT
TREER :RIGHT
PU HOME PD
SETH 90
FD :SIDE/2
SETH 270
```

```
PC 9
FILL :SIDE :STEP :MANY
PU HOME PD
HT
END

TREE 60 60 20 1 4
```

NOTE Save these procedures in a file named RECURSION. Add the following Turtle Tester and Turtle Teaser procedures to the file RECURSION or save them separately under another name.

Turtle Tester. Write a program that uses simple recursion to generate a pattern similar to that which is illustrated in Figure 3-40. (Answer on page 291.)

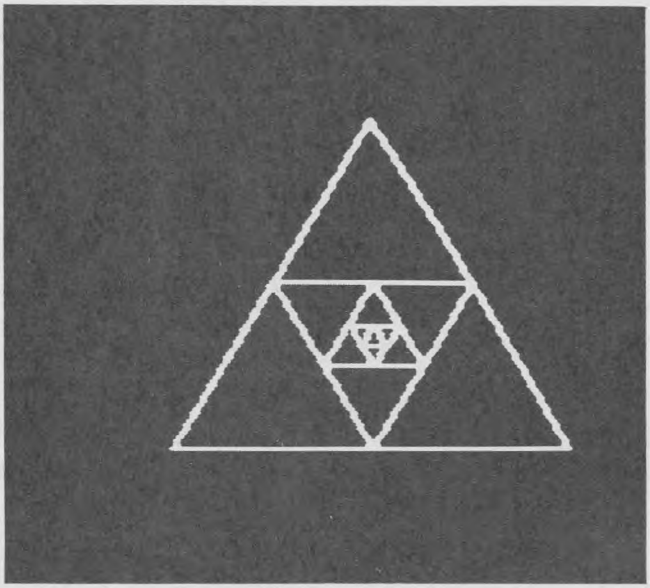


Figure 3-40.

Turtle Teaser. Write a procedure that uses complex recursion to create a design similar to that in Figure 3-41.

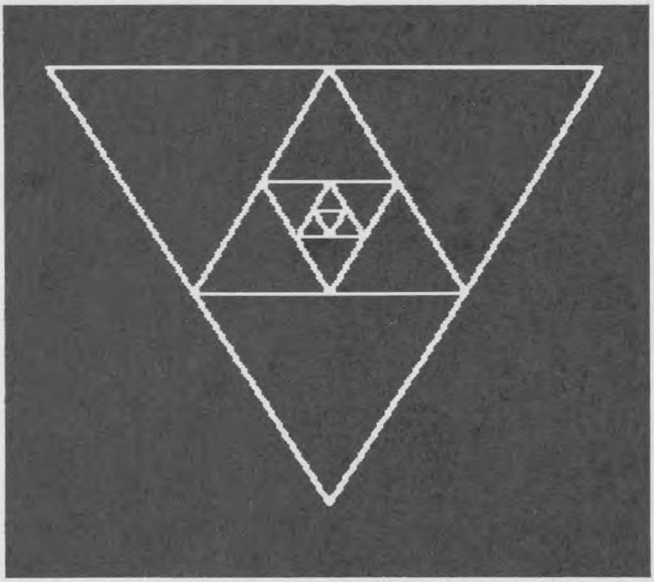


Figure 3-41.

Reference Point 3

Command Review. The following chart contains a summary of all the commands, abbreviations, and definitions introduced in this chapter.

COLOR COMMANDS

PC N Sets the current pen color to the color assigned by the input number N (0 to 15).

BG N Sets the current background screen to the color assigned by the input number N (0 to 15).

SINGLECOLOR Allows the turtle to draw the thinnest possible lines; remaining in effect until DOUBLECOLOR is executed.

DOUBLECOLOR Allows the turtle to draw double thickness but less precise lines.

PC −1 PENERASE erases all dots and lines in the path of the turtle.

SCREEN DISPLAY

NOWRAP The turtle is restricted to moving only within the dimensions of the screen (240 high and 280 wide). Any attempt to move beyond these values results in the error message "TURTLE OUT OF BOUNDS."

WRAP This is the default or normal mode of screen display. The turtle wraps off one edge of the screen and onto the opposite edge. The turtle is always within the screen boundaries.

ARITHMETIC OPERATIONS

x + y Addition is the sum when x is added to y.

x − y Subtraction is the difference when y is subtracted from x.

x * y Multiplication is the product when x is multiplied by y.

x / y Division is the quotient when x is divided by y.

RANDOM x Selects a random non-negative integer between 0 and x − 1, inclusive.

PREDICATES and FUNCTIONS

x = y x equals y.

x < y x is less than y.

x > y x is greater than y.

PRINT COMMANDS

PRINT [. . .] Prints the words or lists contained within the brackets. The abbreviation PR may also be used.

CLEARTEXT Removes all text from the entire text screen. There is no abbreviation for this command.

FLOW OF CONTROL

CO Causes a program interrupted by CTRL-Z to continue.

CTRL-W Stops program execution until any key is pressed.

CTRL-Z Causes the program to stop until CO is typed.

IF CONDITION ACTION1 ACTION2 A conditional statement that checks if the condition is true or false. If the condition is true, ACTION1 is executed. If the statement is false, ACTION2 is executed (either on the same line or the next line).

VARIABLES

:NAME Pronounced "dots name," defines NAME to be a local variable. The contents of NAME is assigned as input through a procedure call and is accessible only to procedures which include it in the procedure heading and definition.

MAKE "NAME DATA Defines NAME to be a global variable where DATA is what is assigned to NAME. The contents of NAME is accessible by any procedure.

Application. Enter the following commands and study the resulting picture. Notice you are loading several previously created files and erasing unwanted procedures from each to accumulate desired procedures in the workspace. You will use these procedures to produce a new picture and save them together in a new file named HOME.SWEET.HOME.

1. Type in the following commands to load the file created in Activity 3.3.

```
READ "COLORFUL.STUFF
POTS
```

Erase all the procedures except the three listed below.

```
SQUARE
PANES
RECTANGLE
```

NOTE Remember, to erase a procedure type:

```
ERASE FLASH2
```

It is not necessary to type the variables that follow the title.

NOTE Type POTS to verify the existence of the three desired procedures in the workspace. Be careful that you do not erase these procedures as you load the next two files.

2. Enter the following commands to load in the file created in Activity 3.4.

```
READ "COLORFUL.POSIES
POTS
```

Erase all the procedures in this file except those listed below. Be careful!

```
POSIE
PETAL
STEM
LEAF
FLOWER
ASSIGN
ARCR
CIRCLER
ARCL
CIRCLEL
ARCL1
ARCR1
CORRECTARCL
CORRECTARCR
```

Your workspace should now contain seventeen procedures.

3. Type in the following commands to load the file created in Activity 3.7.

```
READ "SOLID.PICS
POTS
```

NOTE Erase all the procedures except those listed below.

```
COLORFILL
HOUSE
MOVE
FILL
GLOBALVALUES
```

4. You should now have the following twenty-two procedures in your workspace. To verify that they are correct, compare your procedures to those listed below.

```
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO PANES :SIDE
SQUARE :SIDE/4
SQUARE :SIDE/8
FD :SIDE/8 RT 90
FD :SIDE/8 LT 90
SQUARE :SIDE/8
LT 90 FD :SIDE/8
```

```
RT 90 BK :SIDE/8
END

TO RECTANGLE :LENGTH :WIDTH
REPEAT 2 [FD :LENGTH LT 90 FD :WIDTH LT 90]
END

TO FILL :SIDE
IF :SIDE < :STEP STOP
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :STEP
END

TO COLORFILL :SIDE
PC :COLOR
FILL :SIDE
END

TO MOVE :ANGLE :MORE
PU SETH :ANGLE FD :MORE SETH 0 PD
END

TO HOUSE :SIDE :COLORHOUSE :COLORROOF
PC :COLORHOUSE
GLOBALVALUES 4 :COLORHOUSE 1
RT 90
COLORFILL :SIDE
LT 90
FD :SIDE
PC :COLORROOF
GLOBALVALUES 3 :COLORROOF 3
RT 90
COLORFILL :SIDE
PU LT 90 BK :SIDE PD
END

TO GLOBALVALUES :FIRST :SECOND :THIRD
MAKE "MANY :FIRST
MAKE "COLOR :SECOND
MAKE "STEP :THIRD
END

TO POSIE
```

```
STEM
FLOWER
END

TO ASSIGN
MAKE "COLORSTEM 2
MAKE "COLORFLOWER RANDOM 11

TO PETAL
REPEAT 2 [ARCR 20 60 RT 120]
END

TO FLOWER
PC :COLORFLOWER
REPEAT 8 [PETAL RT 360/8]
PU BK 30
PD
END

TO STEM
PC :COLORSTEM
LT 90
LEAF
RT 90
LEAF
SETH 0
FD 30
END

TO LEAF
REPEAT 2 [ARCR 23 60 RT 120]
END
```

5. Edit PANES to become WINDOW and add the two new procedures as follows:

```
TO WINDOW :SIDE
GLOBALVALUES 4 1 1
RT 90
COLORFILL :SIDE/4
END

TO RAYS
PC 4
```

```
REPEAT 20 [PU FD 20 PD FD 10 PU BK 30 RT 18]
END

TO SUN :SIDE :MANY
PC RANDOM 11
MOVE 90 :MANY
IF :SIDE < 1 STOP
CIRCLER :SIDE
SUN :SIDE - :MANY :MANY
END
```

6. Enter the following procedure. Notice it is the accumulation of the procedures now in your workspace. Compare your screen display to the picture illustrated in Figure 3-42.

```
TO SCENE
DOUBLECOLOR
CLEARTEXT
FULLSCREEN
MOVE 180 75
HOUSE 100 5 4
MOVE 90 75
PC 0
RECTANGLE 48 20
FD 20
CIRCLEL 3
MOVE 270 57
WINDOW 100
MOVE 0 40
WINDOW 100
MOVE 90 40
WINDOW 100
MOVE 135 90
REPEAT 4 [POSIE MOVE 90 40]
MOVE 215 150
SUN 20 5
RAYS
PU HOME PD
SPLITSCREEN
PR [HOME, SWEET, HOME]
HT
END
```

```
SCENE
SAVE "HOME.SWEET.HOME
```

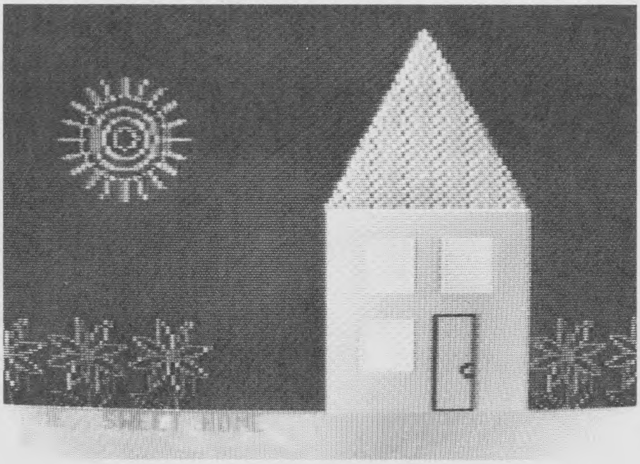
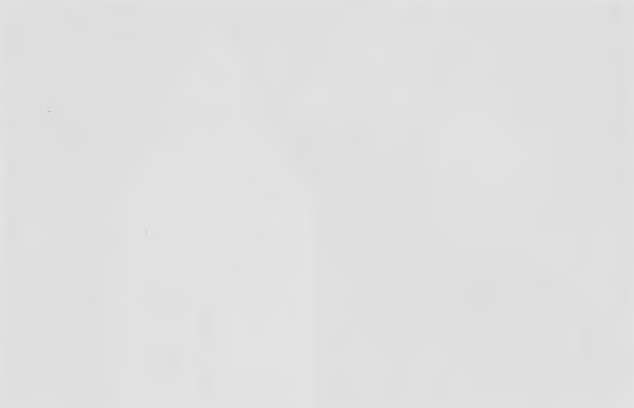


Figure 3-42.

Suggested Projects

1. Add several birds and clouds in the sky.
2. Make the door a different color than the house.
3. Add a chimney to the roof.
4. Make a path leading to the door.
5. Change the color of the house and the roof.

[illegible]

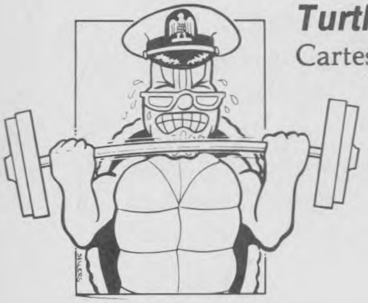


[illegible]

Coordinating Turtle Points

4

Activity 4.1 Turtle Positions



Turtle Trials. This activity introduces the use of (X,Y) Cartesian coordinates to move the turtle faster.

1. The Logo coordinate system is pictured in Figure 4-1. The Roman numerals define the four quadrants of the Cartesian coordinate system. Cartesian coordinates are not entirely new to you if you have plotted points on coordinate paper (graph paper). You can locate a point on graph paper by giving its distance from two perpendicular lines. Perpendicular lines form right angles (90 degrees) with each other. In Figure 4-1, lines L1 and L2 are perpendicular lines that intersect at point A, or the origin. Points on L1 to the right of A are positive and to the left of A are negative. Similarly, points on L2 above A are positive and below A are negative.

   The location of any point in relation to A is identified by an ordered pair (X,Y), where X is the horizontal position in relation to A and Y is the vertical position in relation to A. For example, point W (−10,20) is 10 units to the left of A and 20 units above A. Notice point A (the origin) is identified as (0,0) and is the starting position of the turtle. XCOR gives the horizontal position in relation to A. The vertical position in relation to A is given by YCOR. The purpose of this part is for you to locate various boundary coordinates of the display screen, identified by letters A through I. Enter the following commands and write the ordered pairs in the indicated spaces, both below and on Figure 4-1.

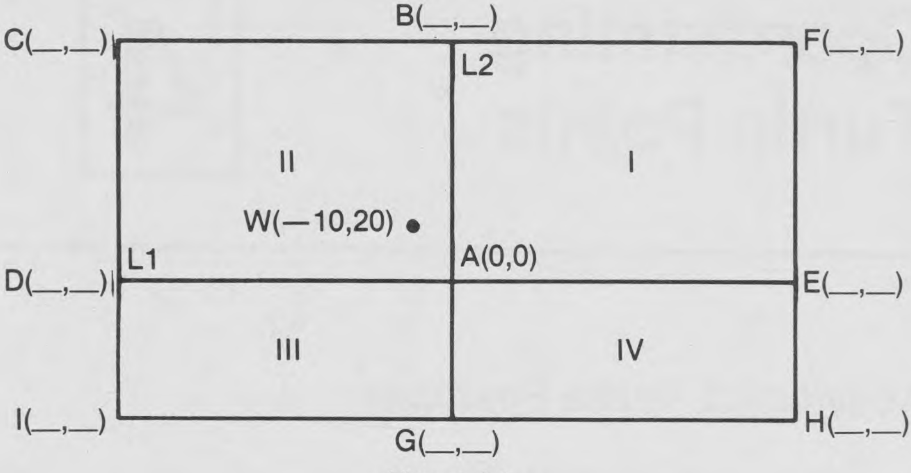


Figure 4-1

```
DRAW
PR XCOR              A(0,0)
PR YCOR
FD 130
XCOR                 B(__,__)
YCOR
LT 90
FD 159
XCOR YCOR
PR XCOR PR YCOR      C(__,__)
LT 90
FD 130
PR XCOR PR YCOR      D(__,__)
LT 90
FD 318
PR XCOR PR YCOR      E(__,__)
LT 90
FD 130
PR XCOR PR YCOR      F(__,__)
LT 90
FD 159
PR XCOR PR YCOR      B(__,__)
LT 90
FD 130
PR XCOR PR YCOR      A(__,__)
FD 76
PR XCOR PR YCOR      G(__,__)
```

```
LT 90
FD 159
PR XCOR PR YCOR  H(__,__)
LT 180
FD 318
PR XCOR PR YCOR  I(__,__)
RT 90
FD 76
PR XCOR PR YCOR  D(____,____)
RT 90
FD 318
PR XCOR PR YCOR  E(__,__)
RT 90
FD 76
PR XCOR PR YCOR  H(__,__)
PU HOME PD
PR XCOR PR YCOR  A(__,__)
```

2. Enter the following commands. Observe the path of the turtle, and fill in the information on the lines provided. Compare the coordinates of the points with those obtained in Part 1.

```
DRAW
PR XCOR PR YCOR  A(0,0)
SETY 130
PR XCOR PR YCOR  B(__,__)
SETX -159
PR XCOR PR YCOR  C(__,__)
SETY 0
PR XCOR PR YCOR  D(__,__)
SETX 159
PR XCOR PR YCOR  E(__,__)
SETY 130
PR XCOR PR YCOR  F(__,__)
SETX 0
PR XCOR PR YCOR  B(__,__)
SETY 0
PR XCOR PR YCOR  A(__,__)
SETY -76
PR XCOR PR YCOR  G(__,__)
SETX 159
PR XCOR PR YCOR  H(__,__)
SETX -159
PR XCOR PR YCOR  I(__,__)
```

```
SETY 0
PR XCOR PR YCOR  D(__,__)
SETX 159
PR XCOR PR YCOR  E(__,__)
SETY -76
PR XCOR PR YCOR  H(__,__)
PU HOME PD
PR XCOR PR YCOR  A(__,__)
```

NOTE The coordinates of the points in Part 1 are the same as those in Part 2.

3. Enter the following commands. Record your observations on the lines following the commands. Compare the coordinates of the points with those in Parts 1 and 2.

```
DRAW
PR XCOR PR YCOR  A(__,__)
SETXY 0 130
PR XCOR PR YCOR  B(__,__)
SETXY -159 130
PR XCOR PR YCOR  C(__,__)
SETXY -159 0
PR XCOR PR YCOR  D(__,__)
SETXY 159 0
PR XCOR PR YCOR  E(__,__)
SETXY 159 130
PR XCOR PR YCOR  F(__,__)
SETXY 0 130
PR XCOR PR YCOR  B(__,__)
SETXY 0 0
PR XCOR PR YCOR  A(__,__)
SETXY 0 -76
SETXY 0 0-76
PR XCOR PR YCOR  G(__,__)
SETXY 159 -76
SETXY 159 (-76)
PR XCOR PR YCOR  H(__,__)
SETXY -159 (-76)
PR XCOR PR YCOR  I(__,__)
SETXY -159 0
PR XCOR PR YCOR  D(__,__)
SETXY 159 0
PR XCOR PR YCOR  E(__,__)
```

```
SETXY 159 (-76)
PR XCOR PR YCOR  H(__,__)
PU HOME PD
PR XCOR PR YCOR  A(__,__)
```

Are the coordinates of points in Part 3 the same as those in Parts 1 and 2? ______________________________

4. Enter the following commands. Record your observations on the lines following the commands. Compare the coordinates of the points with those in Parts 1, 2, and 3.

```
DRAW
PR XCOR PR YCOR           A(0,0)
SETY YCOR+130
PR _______________        B(__,__)
SETX XCOR-159
PR _______________        C(__,__)
SETY YCOR-130
PR _______________        D(__,__)
SETX XCOR+318
PR _______________        E(__,__)
SETY YCOR+130
PR _______________        F(__,__)
SETXY XCOR-159 YCOR+0
PR _______________        B(__,__)
SETXY XCOR+0 YCOR-130
PR _______________        A(__,__)
SETY YCOR-76
PR _______________        G(__,__)
SETX XCOR+159
PR _______________        H(__,__)
SETX XCOR-318
PR _______________        I(__,__)
SETY YCOR+76
PR _______________        D(__,__)
SETXY XCOR+318 YCOR+0
PR _______________        E(__,__)
SETXY XCOR+0 YCOR-76
PR _______________        H(__,__)
PU
SETXY XCOR-159 YCOR+76
PR _______________        A(__,__)
PD
```

5. Load STAMPER from the Utilities Disk. Type the following commands and compare how each command contributes to the partial completion of the screen display illustrated in Figure 4-2. Identify the commands that are incorrect.

```
READ "STAMPER
DRAW
PU SETXY -100 100 PD
SETXY 100 100
SETXY 100 -70
SETXY 100 (-70)
SETXY -100 (-70)
SETXY -100 100
SETXY 100 (-70)
SETXY -100 (-70)
SETXY 100 100
HT
PU SETXY -110 10 STAMPCHAR Y
PU SETXY -110 10 STAMPCHAR "Y
SETXY 70 115 STAMPCHAR "(100,100)
SETXY 70 115 STAMP [(100,100)]
```

6. Enter the following procedure and compare the screen display with the illustration in Figure 4-2.

```
TO FRAME.X
FULLSCREEN
PU SETXY -100 100 PD
SETXY 100 100
SETXY 100 (-70)
SETXY -100 (-70)
SETXY -100 100
SETXY 100 (-70)
SETXY -100 (-70)
SETXY 100 100
PU SETXY -110 10 STAMPCHAR "Y
SETXY 0 (-85) STAMPCHAR "X
SETXY 70 115 STAMP [(100,100)]
SETXY 70 (-85) STAMP [(100,-70)]
SETXY -140 (-85) STAMP [(-100,-70)]
SETXY -140 115 STAMP [(-100,100)]
SETXY -50 (-110) STAMP [A LABELED DIAGRAM]
```

```
HT
END

FRAME.X
f3
```

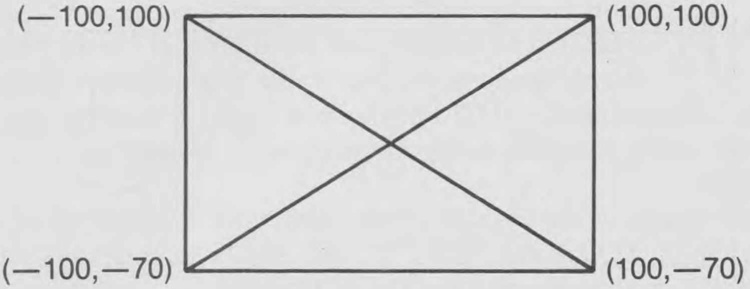


Figure 4-2.

7. Enter the recursive procedures listed below and describe the screen display on the line provided.

```
DRAW
TO EXPLOSION :NUMBER
HT
IF :NUMBER = 0 STOP
PU HOME PD
MAKE "SIGN RANDOM 2
IF :SIGN = 0 MAKE "SIGN (-1)
MAKE "XPOS (RANDOM 130)*:SIGN
MAKE "YPOS RANDOM 100
SETXY :XPOS :YPOS
EXPLOSION :NUMBER - 1
PU SETXY -35 (-70) STAMP [EXPLOSION!!!]
END

EXPLOSION 70 ________________________________
DRAW
EXPLOSION 100 _______________________________
```

Turtle Talk. Commodore Logo permits the turtle to be positioned anywhere on the screen by using Cartesian coordinates. Figure 4-3 summarizes the screen coordinates that should have been obtained during the activity. Each coordinate is defined as an ordered pair, where the first number indicates the horizontal position (X) and the second number indicates the vertical position (Y). Both are required to identify each point uniquely in the two-dimensional space in relation to the origin. Notice the center of the screen is defined to be (0,0). Positive and negative numbers are used to indicate the other positions using the coordinate system typically used in mathematics.

The coordinates of the position of the turtle may be observed at any time by using PRINT XCOR and PRINT YCOR. XCOR indicates the horizontal position (or X coordinate) and the YCOR identifies the vertical position (or Y coordinate). The PRINT command is not required before either XCOR or YCOR. In this case Logo responds with RESULT: N, where N is a screen coordinate. However, Part 1 demonstrates that PRINT must accompany both XCOR and YCOR when requested on the same line.

There are several methods by which the turtle can be located on the screen. First, the FD and BK commands used in Part 1 serve as review. In Part 2, SETX and SETY position the turtle in either a horizontal or vertical direction. SETXY, which is introduced in Part 3, places the turtle at the coordinates of a position that is immediately obvious by the values included in the command. For example, SETXY 100 50 moves the turtle to a point where the (X,Y) coordinate is (100,50). You must be careful to avoid ambiguity when writing coordinates with negative values. For example, SETXY 0 -76 results in an error message because Logo does not know whether to interpret 0 -76 as subtraction or negative input. This confusion is easily corrected by using SETXY 0 0-76 or SETXY 0 (-76).

Part 4 demonstrates another version of the use of SETX, SETY, and SETXY. Here turtle movement from one coordinate to the next is accomplished by adding or subtracting a value to the previous coordinate. This position change can occur horizontally, vertically or diagonally (both horizontally and vertically).

Part 5 introduces how letters can be displayed on the graphics screen. The STAMPCHAR command puts one character at a time on the screen. This character must be preceded by a quotation mark("). Words and messages can also be displayed, but STAMPER, a file on the Utilities Disk, must be read into memory first. STAMPER contains STAMP, a procedure that prints words or a list of words at any location on the graphics screen. The message to be printed must be enclosed with brackets ([ ]). A combination of the use of STAMPCHAR and STAMP is illustrated in Part 6 to highlight, in letters, important features of the diagram.

Part 7 presents a procedure that draws lines from the origin to some randomly generated point (XPOS, YPOS). This procedure shows that SETXY accepts variables as input, as well as constant numerical values.

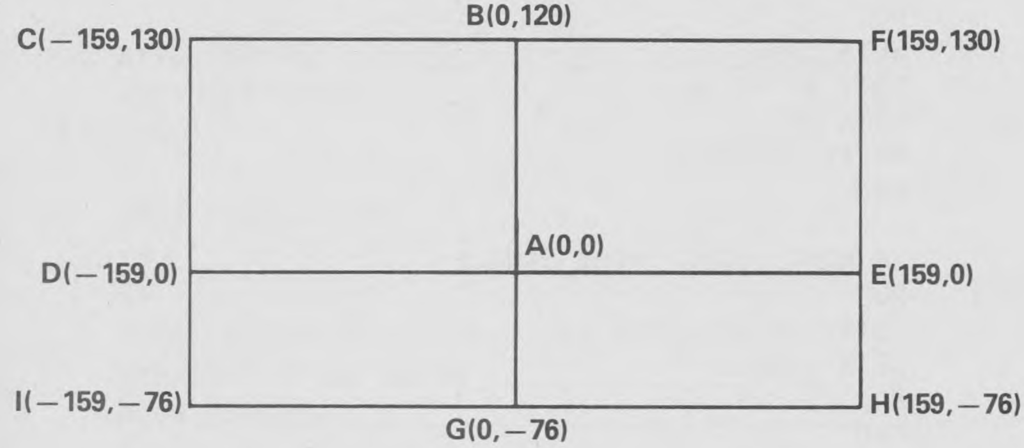


Figure 4-3.

Turtle Truth. Study the procedures and inputs, and predict their outcome. Type in the commands and observe the results.

```
DRAW
TO AXIS :XPOS :YPOS
HT
LONG :YPOS
LAT :XPOS
PU SETXY 10 (-10) STAMP [(0,0)]
SETXY 0 (-1*:YPOS + 8) STAMPCHAR "Y
SETXY (-1*:XPOS + 5) 0 STAMPCHAR "X
END

TO LONG :YPOS
PU
```

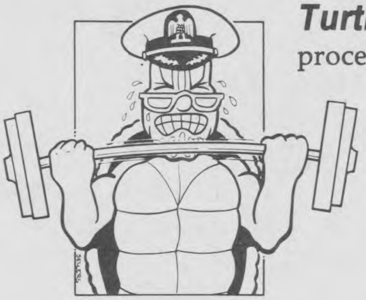
```
SETX 0
SETY :YPOS
PD FD :YPOS*-2
END

TO LAT :XPOS
PU
SETX :XPOS
SETY 0
SETH 90
PD FD :XPOS*-2
END

TO MOVE :XPOS :YPOS :ANGLE
PU
SETXY :XPOS :YPOS
SETH :ANGLE
PD
END

TO LBRACKET :SIDE :XPOS :YPOS :ANGLE
MOVE :XPOS :YPOS :ANGLE
FD :SIDE
SETH 0
FD :SIDE*2
SETH 90
FD :SIDE
END

TO RBRACKET :SIDE :XPOS :YPOS :ANGLE
MOVE :XPOS :YPOS :ANGLE
FD :SIDE
SETH 0

FD :SIDE*2
SETH 270
FD :SIDE
END

TO POSITIVE :SIDE
PD REPEAT 4 [FD :SIDE BK :SIDE RT 90]
END
```

```
TO NEGATIVE :SIDE
PD
SETH 90
FD :SIDE*2
END

AXIS -60 -60 ______________________________
AXIS -60 (-60) ____________________________
DRAW

AXIS -149 (-120) __________________________
f5
f3
LBRACKET 10 30 40 270 _____________________
RBRACKET 10 80 40 90 ______________________
MOVE 40 50 0 ______________________________
POSITIVE 10 _______________________________
MOVE 65 50 0 ______________________________
POSITIVE 10 _______________________________
```

Turtle Tester. Use the procedures provided in the Turtle Truth to label each quadrant of the coordinate axis system with the appropriate positive or negative symbols indicating the correct coordinate signs. One quadrant has been done for you in the Turtle Truth. Save your work under the file name GRID. (Answer on page 291.)

Turtle Teaser. Use the procedures LBRACKET, RBRACKET, POSITIVE, and NEGATIVE to produce one master procedure that draws and labels all of the quadrants.

Activity 4.2 Turtle Placement



Turtle Trials. This activity incorporates variables in procedures to draw a diagram anywhere on the screen.

1. Enter the following procedure to draw the frame of a kite as illustated in Figure 4-4. Observe the drawing on the screen.

```
TO FRAME1
SPLITSCREEN
PU
SETXY 50 40
PD PC 5
SETXY 50 90
SETXY 30 70
SETXY 50 40
SETXY 70 70
SETXY 50 90
SETXY 70 70
SETXY 30 70
PU
SETXY 50 40
PD
END
FRAME1 ____________________________________
```

2. Create FRAME2 by editing FRAME1 to read as listed below.

```
TO FRAME2
SPLITSCREEN
PU
SETXY -60 (-30)
PD PC 5
SETXY -60 20
```

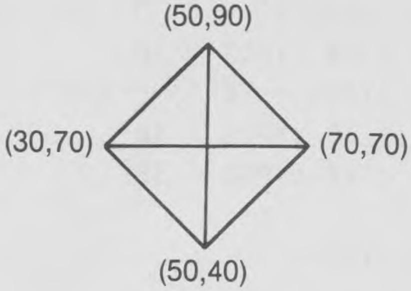


Figure 4-4.

```
SETXY -80 0
SETXY -60 (-30)
SETXY -40 0
SETXY -60 20
SETXY -40 0
SETXY -80 0
PU
SETXY -60 (-30)
PD
END
```

FRAME1 ____________________________________

FRAME2 ____________________________________

Compare and describe the shape and position of the kite drawn by FRAME1 with the kite drawn in FRAME2. ____________________

Describe how the coordinates used in each procedure are similar. ______

____

3. Create FRAME by editing FRAME1 so it reads as follows. Figure 4-5 identifies the new form of (X,Y) coordinates.

```
TO FRAME :XPOS :YPOS :COLORFRAME
FULLSCREEN
PU
SETXY :XPOS :YPOS
PD PC :COLORFRAME
SETXY :XPOS :YPOS + 50
SETXY :XPOS - 20 :YPOS + 30
SETXY :XPOS :YPOS
SETXY :XPOS + 20 :YPOS + 30
SETXY :XPOS :YPOS + 50
SETXY :XPOS + 20 :YPOS + 30
SETXY :XPOS - 20 :YPOS + 30
PU
SETXY :XPOS :YPOS
PC 1
END
```

```
FRAME 50 40 5 ______________________________
FRAME -60 (-30) 3 ___________________________
```

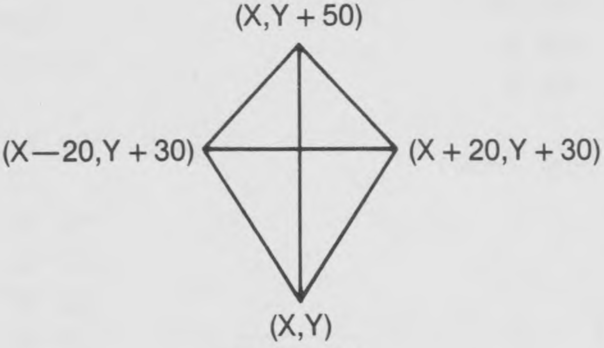


Figure 4-5.

4. Create FRAME3 by editing FRAME once more so it reads as follows. Note the change of (X,Y) coordinates to agree with the illustration in Figure 4-6.

```
TO FRAME3 :XPOS :YPOS :COLORFRAME
FULLSCREEN
PU
SETXY :XPOS :YPOS - 50
PD PC :COLORFRAME
SETXY :XPOS :YPOS
SETXY :XPOS - 20 :YPOS - 20
SETXY :XPOS :YPOS - 50
SETXY :XPOS + 20 :YPOS - 20
SETXY :XPOS :YPOS
SETXY :XPOS + 20 :YPOS - 20
SETXY :XPOS - 20 :YPOS - 20
PU
SETXY :XPOS :YPOS - 50
PC 1
END

FRAME3 50 40 5 ______________________________
f3
FRAME3 -85 50 4 _____________________________
f3
FRAME3 -2 (-15) 3 ___________________________
f3
FRAME3 -75 (-50) 2 __________________________
f3
```

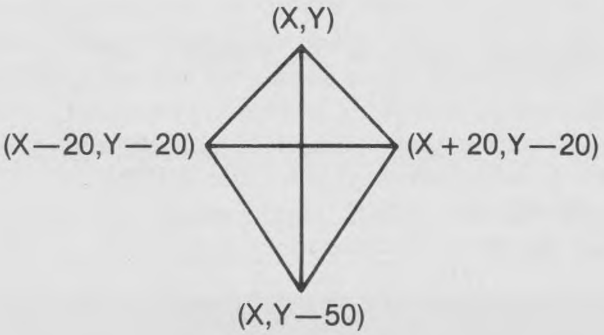


Figure 4-6.

5. Let's add a tail to the kite in definition mode. Notice the action of the turtle as you push return after the SETH TOWARDS command.

```
DRAW
FRAME 50 40 5
PD
SETH TOWARDS -10 (-10)
FD 24
SETH TOWARDS 5 (-20)
FD 8
SETH TOWARDS 40 (-50)
FD 10
SETH TOWARDS 100 (-80)
FD 30
```

NOTE The pair of numbers following SETH TOWARDS are the coordinates of the position to which the turtle points.

6. Now let's define the following procedure to draw a tail for the kite illustrated in Figure 4-7, but without the bows.

```
TO TAIL :XPOS :YPOS :COLORTAIL
PD
PC :COLORTAIL
SETH TOWARDS :XPOS - 10 :YPOS - 10
FD 16
SETH TOWARDS :XPOS - 16 :YPOS - 20
FD 8
SETH TOWARDS :XPOS - 14 :YPOS - 20
FD 8
SETH TOWARDS :XPOS - 7 :YPOS - 27
FD 16
SETH TOWARDS :XPOS + 24 :YPOS - 49
FD 16
END
```

```
FRAME 50 40 5 ______________________________
TAIL 50 40 3 _______________________________
```

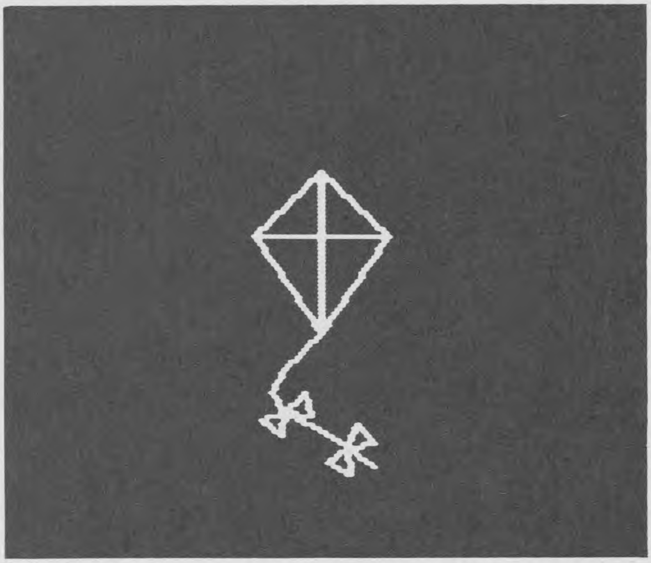


Figure 4-7.

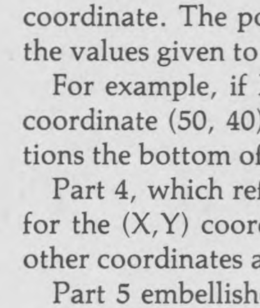
Turtle Talk. Parts 1 and 2 each draw a picture of the frame of a kite, but at different locations on the screen. The procedures FRAME1 and FRAME2 differ only in the coordinates for the SETXY command. Fortunately, there is a better way to display a diagram in a variety of locations on the screen other than this cumbersome approach.

Part 3 introduces a method whereby the coordinates for the SETXY command are generalized by using variables, rather than specific numerical values. Figure 4-5 shows that the (X,Y) coordinate is located at the bottom of the kite frame. Other coordinate values are determined by their position in relation to the (X,Y) coordinate. The position of the displayed kite on the screen depends upon the values given to X and Y.

For example, if X = 50 and Y = 40, the bottom of the kite is placed at coordinate (50, 40) on the screen. An (X,Y) coordinate of (−60, −30) positions the bottom of the kite on the screen where X = −60 and Y = −30.

Part 4, which refers to Figure 4-6, demonstrates that the selected position for the (X,Y) coordinate (top of the kite frame) is arbitrary, assuming the other coordinates are modified accordingly.

Part 5 embellishes the kite by adding a tail. Notice the (X,Y) coordinate of TAIL (refer to Figure 4-7) is defined at the top of the tail to assure that FRAME and TAIL are properly coordinated to appear as one connected picture. The SETH TOWARDS command turns the turtle so it faces toward a designated point on the display screen. Part 5 demonstrates how you can experiment with the drawing of more intricate designs in definition mode. Part 6 shows how variables can be used with SETH TOWARDS to create irregular designs.



Turtle Truth. Enter the procedures as listed. Study the inputs and predict the position as determined by the inputs for each variable.

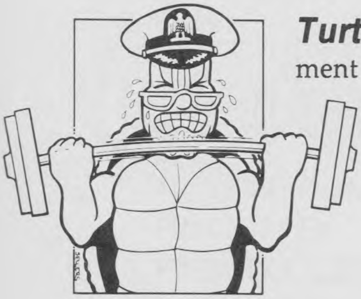
```
TO NIGHT.KITE :NUMBER :XPLACE :YPLACE
 :COLORFRAME :COLORTAIL
STARS :NUMBER
FRAME :XPLACE :YPLACE :COLORFRAME
TAIL :XPLACE :YPLACE :COLORTAIL
```

```
BOW
END

TO STARS :NUMBER
IF :NUMBER = 0 STOP
FULLSCREEN
HT
MAKE "SIGN RANDOM 2
IF :SIGN = 0 MAKE "SIGN (-1)
MAKE "XPOS (RANDOM 159)*:SIGN
MAKE "SIGN RANDOM 2
IF :SIGN = 0 MAKE "SIGN (-1)
MAKE "YPOS (RANDOM 130)*:SIGN
STAR :XPOS :YPOS
STARS :NUMBER - 1
END

TO STAR :XPOS :YPOS
PU SETXY :XPOS :YPOS PD
SETH 0
REPEAT 5 [FD 3 BK 3 RT 72]
END

TO BOW
PC 1
RT 45
REPEAT 3 [FD 8 RT 120]
LT 90
REPEAT 3 [FD 8 LT 120]
END

NIGHT.KITE 5 50 40 3 2 ____________________
DRAW
f3
NIGHT.KITE 10 (-80) (-30) 2 4 ____________
DRAW
f3
```

Turtle Tester. Create a procedure to draw a sky full of randomly colored kites. Make each tail have three bows. Save it as a file named SKYFULL.OF.KITES. (Answer on page 294.)

Turtle Teaser. Design a box kite using the commands introduced in Activities 4.1 and 4.2. It may be helpful to draw the design and plot the coordinates on graph paper before working in toplevel mode or defining the procedure itself.

Activity 4.3 Turtle Tessellations



Turtle Trials. This activity demonstrates the development of fullscreen tessellations.

1. Enter the following commands and observe the pattern that is created.

```
TO SQUARE :SIDE
FULLSCREEN
MAKE "COLOR (1 + RANDOM 5)
PC :COLOR
REPEAT 4 [FD :SIDE RT 90]
FD :SIDE
END
```

```
PR 260/26 ______________________________

REPEAT 10 [SQUARE 26] __________________

f3
DRAW
PR 260/15 ______________________________

REPEAT 17.3333 [SQUARE 15] _____________
f3
DRAW

PR 260/52 ______________________________
REPEAT 5 [SQUARE 52] ___________________
f3
DRAW
```

2. Enter the procedures and observe the results.

```
TO VROW :LENGTH :XPLACE :YPLACE
FULLSCREEN
PU SETXY :XPLACE :YPLACE PD
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER*(-1)
REPEAT :NUMBER [SQUARE :LENGTH]
END

VROW 30 (-140) (-120) __________________
f3
DRAW

VROW 19 (-139) (-119) __________________
f3
DRAW

VROW 7 (-130) (-110) ___________________
f3
DRAW
```

3. Edit the procedure to read as follows and observe the results.

```
TO VROW :LENGTH :XPLACE :YPLACE
FULLSCREEN
PU SETXY :XPLACE :YPLACE PD
MAKE "POSX :XPLACE
MAKE "POSY :YPLACE
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER*(-1)
REPEAT :NUMBER [SQUARE :LENGTH]
PU SETXY :POSX :POSY PD
END

VROW 30 (-140)(-120) ________________________
f3
PR :NUMBER
DRAW

VROW 45 (-159)(-128) ________________________
f3
PR :NUMBER
DRAW
```

4. Edit VROW to read as follows. The INTEGER command truncates (removes the decimal part of the number) for positive numbers.

```
TO VROW :LENGTH :XPLACE :YPLACE
FULLSCREEN
PU SETXY :XPLACE :YPLACE PD
MAKE "POSX :XPLACE
MAKE "POSY :YPLACE
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER*(-1)
MAKE "NUMBER INTEGER :NUMBER
REPEAT :NUMBER [SQUARE :LENGTH]
PU SETXY :POSX :POSY PD
END

VROW 45 (-159)(-128) ________________________
```

```
f3
PR :NUMBER ______________________________
```

How does the value for NUMBER compare for that obtained in Part 3? _

____

5. Enter the following procedure and observe the screen display.

```
TO MOVE :LENGTH
PU RT 90 FD :LENGTH LT 90 PD
END

VROW 45 (-159)(-128) _____________________
MOVE 45 __________________________________
VROW 45 (-159 + 45) (-119) _______________
DRAW
```

6. Enter the following procedure and record your observations.

```
TO ACROSS :LENGTH :XPLACE :YPLACE
VROW :LENGTH :XPLACE :YPLACE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE
END

ACROSS 45 (-159)(-128) ___________________
```

Use CTRL-G to stop the procedure when it begins to wrap around.

```
DRAW
```

7. Modify the ACROSS procedure to read as follows and observe the results.

```
TO ACROSS :LENGTH :XPLACE :YPLACE
IF :XPLACE > 159 - :LENGTH STOP
VROW :LENGTH :XPLACE :YPLACE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE
END

ACROSS 45 (-159)(-128) ___________________
f3
```

```
DRAW

ACROSS 20 (-130)(-110) ____________________
f3
DRAW

ACROSS 10 (-135)(-115) ____________________
f3
DRAW
SAVE "FIRST.TESS
```

Turtle Talk. A tessellation is a careful arrangement of parts into a coherent pattern or picture. Tessellations are of particular interest to mathematicians because they demonstrate the geometric relationship of mathematics to art. Since tessellations essentially fill space in a repetitive nature, the computer becomes a useful tool for the creation of tessellations.

The first step is to decide upon some design to be tessellated. To begin, this activity tessellates a simple square, as defined in Part 1.

With Turtle Graphics, tessellations are two-dimensional designs. Therefore, the repetitive drawing of the diagram must be controlled vertically and horizontally, and be symmetrically coordinated to form a mosaic-type design.

Parts 2, 3 and 4 develop a procedure to control the vertical positioning of the design. Once a column design is completed, conditions must be established for drawing in an adjacent column. The MOVE procedure defined in Part 5 accomplishes this task. ACROSS is developed in Part 6 and controls the horizontal drawing of the tessellation, to avoid screen wrap-around.

The Turtle Truth example tessellates a more complex design, our previously created turtle. Several of the procedures used for tessellating the square are reused to tessellate the turtle. The original turtle is redefined using variables to control its size. The turtle design is assigned a square area of space, then tessellated in much the same way as a square of that size.

Turtle Truth. Enter the following procedures to create a new tessellation. Notice the object used in the tessellation is contained in a SQUARE, and can be plugged into the VROW and ACROSS procedures to create a new tessellation. Compare your final screen display to the illustration in Figure 4-8.

```
TO TURTLE :DISTANCE
START :DISTANCE
SHELL :DISTANCE
FORM :DISTANCE
PAWS :DISTANCE
TAIL :DISTANCE
HEAD :DISTANCE
RETURN :DISTANCE
END

TO START :DISTANCE
PU RT 90 FD :DISTANCE*1.5 LT 90 PD
END

TO BOX :DISTANCE
REPEAT 4 [FD :DISTANCE RT 90]
END

TO SHELL :DISTANCE
REPEAT 2 [BOX :DISTANCE FD :DISTANCE]
BOX :DISTANCE
BK :DISTANCE
LT 90
BOX :DISTANCE
BK :DISTANCE*2
BOX :DISTANCE
END

TO SHAPE :DISTANCE
RT 90
REPEAT 4 [FD :DISTANCE LT 45 FD :DISTANCE*1.4
 LT 45]
LT 90
END
```

```
TO PAWS :DISTANCE
LT 45
FD :DISTANCE*.4
LT 90
REPEAT 4 [ARCR :DISTANCE/3 180 LT 90
 FD :DISTANCE*.4 RT 45 FD :DISTANCE RT 45
 FD :DISTANCE*.4 LT 90]
END

TO TAIL :DISTANCE
RT 90
FD :DISTANCE
RT 45
FD :DISTANCE*.4
TRI :DISTANCE/3 120
END

TO TRI :DISTANCE :ANGLE
REPEAT 3 [FD :DISTANCE LT :ANGLE]
END

TO HEAD :DISTANCE
BK :DISTANCE*.4
RT 90
FD :DISTANCE*3
LT 90
REPEAT 3 [FD :DISTANCE RT 120]
END

TO RETURN :DISTANCE
FD :DISTANCE
RT 90
BK :DISTANCE*3
PU LT 90 FD :DISTANCE*1.5 RT 90 PD
END

TURTLE 10

SQUARE 4*10
f3
```

Edit the existing procedures MOVE, VROW, and ACROSS to read as follows:

```
TO MOVE :LENGTH
PU FD :LENGTH PD
END

TO VROW :LENGTH :XPLACE :YPLACE :DISTANCE
FULLSCREEN
PU SETXY :XPLACE :YPLACE PD
MAKE "POSX :XPLACE
MAKE "POSY :YPLACE
MAKE "NUMBER (2*:YPLACE)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER*(-1)
MAKE "NUMBER INTEGER :NUMBER
REPEAT :NUMBER [TURTLE :DISTANCE MOVE :LENGTH]
PU SETXY :POSX :POSY PD
END

VROW 40 (-159)(-128) 10
DRAW

TO ACROSS :LENGTH :XPLACE :YPLACE :DISTANCE
IF :XPLACE > 159 - :LENGTH STOP
VROW :LENGTH :XPLACE :YPLACE :DISTANCE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE
 :DISTANCE
HT
END

ACROSS 40 (-140)(-120) 10
f3
SAVE "TURTLE.TESS
DRAW
```

Turtle Tester. Study the illustration in Figure 4-9 and enter the following procedures used to create STAR. Modify the VROW, MOVE, and ACROSS procedures to create a fullscreen tessellation with STAR. (Answer on page 295.)

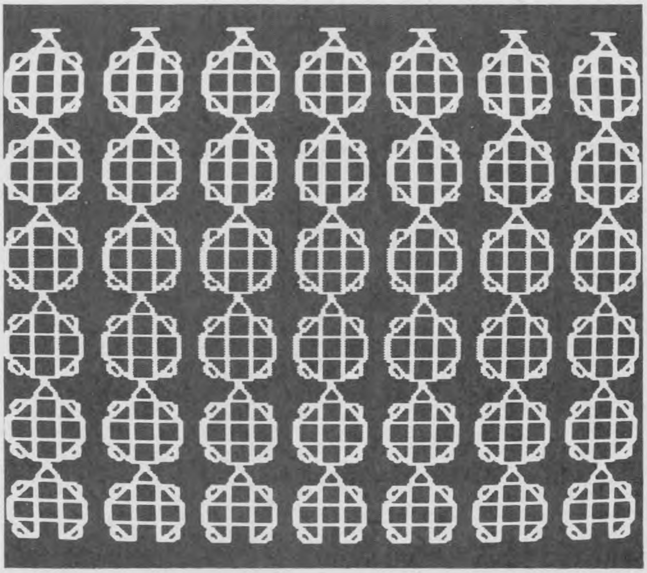


Figure 4-8.

```
TO STAR :DISTANCE :MANY
COLOR
REPEAT :MANY [VARICUBE :DISTANCE RT 360/:MANY]
END

TO COLOR
PC RANDOM 11
END

TO SIDE :DISTANCE
RT 45 FD :DISTANCE
LT 45 FD :DISTANCE
LT 135 FD :DISTANCE
LT 45 FD :DISTANCE
RT 180
END

TO VARICUBE :DISTANCE
BOX :DISTANCE
SIDE :DISTANCE
RT 90 FD :DISTANCE LT 90
SIDE :DISTANCE
LT 90 FD :DISTANCE RT 135 FD :DISTANCE LT 45
BOX :DISTANCE
```

```
RT 45 BK :DISTANCE LT 45
END

STAR 10 8
```

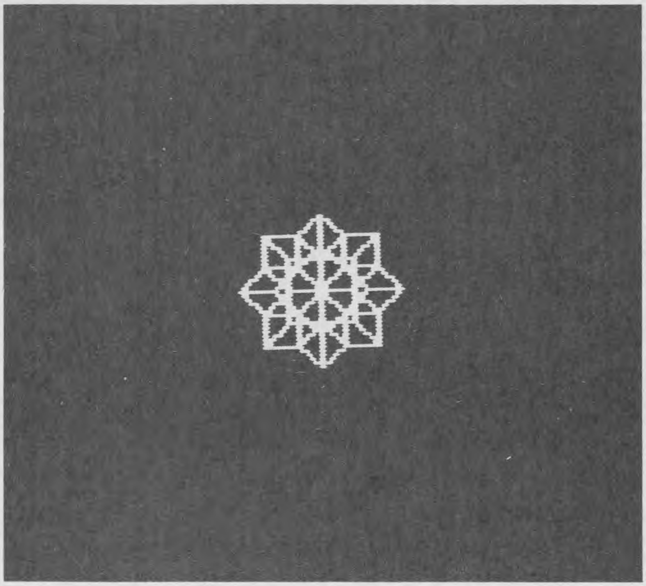


Figure 4-9.

NOTE After you have successfully tessellated STAR, save all the necessary procedures in a file named STAR.TESS.

Turtle Teaser. Create your own object and use it in a fullscreen tessellation. Save this tessellation under a file name of your choice.

Reference Point 4

Command Review. This section introduces the definitions of terms and commands presented in Chapter 4.

Terms

Screen Coordinates The turtle graphics screen is divided into small entities called dots or pixels. There are approximately 318 columns and 248 rows of pixels where the turtle is in view. Each pixel has a unique identification known as a screen coordinate. Figure 4-10 identifies several screen coordinates and indicates their relative position on the display screen.

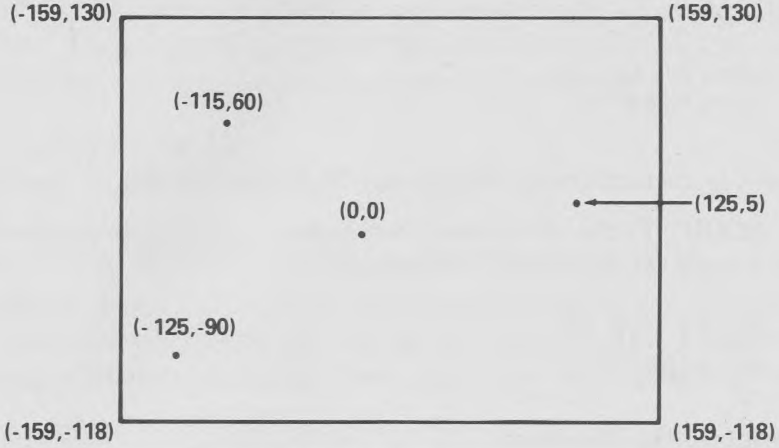


Figure 4-10.

Ordered Pairs Each screen coordinate is described as an ordered pair, where the first number indicates the horizontal position (X) and the second number indicates the vertical position (Y). The ordered pair (0,0) is located at the center of the screen. Positive and negative numbers are used in the ordered pairs to indicate the relative position of the point to (0,0). It is common to refer to an ordered pair as an (X,Y) coordinate where X means horizontal direction and Y means vertical direction.

Tessellation A computer is particularly useful when repetitive processes are required. A tessellation is a design that results from repeatedly drawing a picture in a coordinated and consistent manner to form a mosaic-like pattern.

Commands

SETXY N1 N2 Moves the turtle to the screen coordinate identified by the ordered pair (N1,N2) where N1 and N2 are integer constants. N2 is enclosed by parentheses when negative. For example: SETXY -50 40 moves the turtle to coordinate (-50,40). SETXY -60 (-70) moves the turtle to coordinate (-60, -70).

SETX N1 Moves the turtle horizontally to the X-coordinate specified by N1, where N1 is an integer input constant. For example:

```
SETXY -50 40
SETX 50
```

moves the turtle from coordinate (-50,40) to (50,40).

SETY N2 Moves the turtle vertically to the Y-coordinate specified by N2, where N2 is an integer input constant. For example:

```
SETXY -50 40
SETY -60
```

moves the turtle from coordinate (-50,40) to (-50,-60).

PR XCOR Prints the current X-coordinate (or horizontal position) of the turtle on the screen. For example:

```
SETXY -50 40
PR XCOR
```

prints -50 on the screen.

PR YCOR Prints the current Y-coordinate (or vertical position) of the turtle on the screen. For example:

```
SETXY -50 40
PR YCOR
```

prints 40 on the screen. XCOR and YCOR can be used without PR. For example:

```
XCOR
```

prints **RESULT:** -50 on the screen.

```
YCOR
```

prints **RESULT:** 40 on the screen.

SETH TOWARDS N1 N2 Points the tip of the turtle to the screen coordinate (N1,N2), where N1 and N2 are input integer constants. For example:

```
SETH TOWARDS -100 50
```

points the turtle towards the screen coordinate (-100,50).

STAMPCHAR Prints one character at a time on the graphics screen. For example,

```
PU SETXY 0 0
STAMPCHAR "B
```

prints a B in the HOME position.

STAMP Is a procedure in STAMPER, a file on the Utilities Disk, which prints a word or list on the graphics screen. For example,

```
PU SETXY -50 50
STAMP [LOGO IS FUN]
```

prints **LOGO IS FUN** beginning a screen coordinate (-50,50).

Application. Enter the following procedures and study the resulting display. Identify the procedures controlling the border and the positioning of the message. Compare your final screen display to the illustration in Figure 4-11.

```
TO SECTION :LENGTH :STEP :MANY :DISTANCE
MAKE "POSX XCOR
MAKE "POSY YCOR
IF :LENGTH < :STEP STOP
REPEAT :MANY [FD :LENGTH RT 360/:MANY]
SECTION :LENGTH - :STEP :STEP :MANY
 :DISTANCE
SETXY :POSX :POSY
PU FD :DISTANCE PD
END

TO HROW :LENGTH :STEP :MANY :XPOS :YPOS
 :DISTANCE
PU SETXY :XPOS :YPOS PD
MAKE "POSX XCOR
MAKE "POSY YCOR
MAKE "NUMBER (2*:XPOS)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER*(-1)
MAKE "NUMBER INTEGER :NUMBER
REPEAT :NUMBER [SECTION :LENGTH :STEP :MANY
 :DISTANCE]
PU SETXY :POSX :POSY PD
```

```
END

TO VROW :LENGTH :STEP :MANY :XPOS :YPOS
 :DISTANCE
PU SETXY :XPOS :YPOS PD
MAKE "POSX XCOR
MAKE "POSY YCOR
MAKE "NUMBER (2*:YPOS)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER*(-1)
MAKE "NUMBER INTEGER :NUMBER
REPEAT :NUMBER [SECTION :LENGTH :STEP :MANY
 :DISTANCE]
PU SETXY :POSX :POSY PD
END

TO FRAME
BG 1
PC 5
VROW 15 1.5 4 (-139) (-110) 15
FD 15 RT 90 FD 15
HROW 15 1.5 4 (-139) 110 15
FD 15 RT 90 FD 15
VROW 15 1.5 4 139 110 15
FD 15 RT 90 FD 15
HROW 15 1.5 4 139 (-110) 15
HT
END

TO SAMPLER
FULLSCREEN
HT
FRAME
H -80 40 5 7 3
O -40 40 4 7 3
M 0 40 2 7
E 40 40 3 7 4 3
S -100 (-20) 3 2 5
W -60 (-20) 3 7
E -20 (-20) 5 7 4 3
E 20 (-20) 4 7 4 3
T 60 (-20) 2 6 5
H -80 (-80) 4 7 3
O -40 (-80) 3 7 3
```

```
M 0 (-80) 2 7
E 40 (-80) 5 7 4 3
PU PC 0 SETXY -70 (-90) STAMP [A BRADY PRODUCTION]
END

TO W :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
CROSS :XPLACE + 6 :YPLACE + 6 :COLOR
CROSS :XPLACE + 12 :YPLACE + 12 :COLOR
CROSS :XPLACE + 18 :YPLACE + 6 :COLOR
END

TO M :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
CROSS :XPLACE + 6 :YPLACE + 30 :COLOR
CROSS :XPLACE + 12 :YPLACE + 24 :COLOR
CROSS :XPLACE + 18 :YPLACE + 30 :COLOR
END

TO S :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
HSIDEN2 :XPLACE :YPLACE :COLOR :LENGTH2
HSIDEN2 :XPLACE :YPLACE + 18 :COLOR :LENGTH2
HSIDEN2 :XPLACE :YPLACE + 36 :COLOR :LENGTH2
VSIDEN1 :XPLACE :YPLACE + 24 :COLOR :LENGTH1
VSIDEN1 :XPLACE + 24 :YPLACE + 6 :COLOR
 :LENGTH1
END

TO T :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
VSIDEN1 :XPLACE + 12 :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE :YPLACE + 36 :COLOR :LENGTH2
END

TO E :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
 :LENGTH3
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE + 6 :YPLACE :COLOR :LENGTH2
HSIDEN2 :XPLACE + 6 :YPLACE + 18 :COLOR
 :LENGTH3
HSIDEN2 :XPLACE + 6 :YPLACE + 36 :COLOR
 :LENGTH2
```

```
END

TO VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
IF :LENGTH1 = 0 STOP
CROSS :XPLACE :YPLACE :COLOR
VSIDE7 :XPLACE :YPLACE + 6 :COLOR
 :LENGTH1 - 1
END

TO HSIDEN2 :XPLACE :YPLACE :COLOR :LENGTH2
IF :LENGTH2 = 0 STOP
CROSS :XPLACE :YPLACE :COLOR
HSIDE3 :XPLACE + 6 :YPLACE :COLOR
 :LENGTH2 - 1
END

TO O :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE + 6 :YPLACE :COLOR :LENGTH2
HSIDEN2 :XPLACE + 6 :YPLACE + 36 :COLOR
 :LENGTH2
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
END

TO H :XPLACE :YPLACE :COLOR :LENGTH1 :LENGTH2
VSIDEN1 :XPLACE :YPLACE :COLOR :LENGTH1
HSIDEN2 :XPLACE + 6 :YPLACE + 18 :COLOR
 :LENGTH2
VSIDEN1 :XPLACE + 24 :YPLACE :COLOR :LENGTH1
END

TO HSIDE3 :XPLACE :YPLACE :COLOR :LENGTH2
IF :LENGTH2 = 0 STOP
CROSS :XPLACE :YPLACE :COLOR
HSIDE3 :XPLACE + 6 :YPLACE :COLOR
 :LENGTH2 - 1
END

TO VSIDE7 :XPLACE :YPLACE :COLOR :LENGTH1
IF :LENGTH1 = 0 STOP
CROSS :XPLACE :YPLACE :COLOR
VSIDE7 :XPLACE :YPLACE + 6 :COLOR
 :LENGTH1 - 1
```

```
END

TO CROSS :XPLACE :YPLACE :COLOR
PC :COLOR
PU SETXY :XPLACE :YPLACE
PD
SETXY LIST :XPLACE + 5 :YPLACE + 5
PU SETXY LIST :XPLACE + 5 :YPLACE
PD
SETXY :XPLACE :YPLACE + 5
END

SAMPLER
```

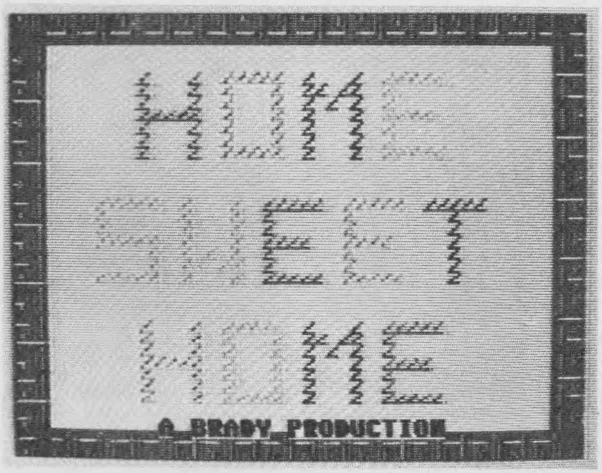


Figure 4-11.

```
SAVE "DOMESTIC
```

Suggested Projects

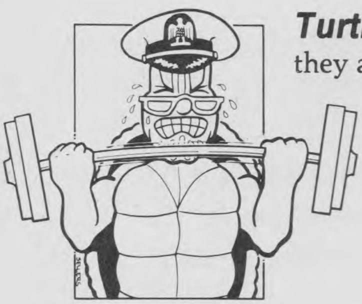
1. Change the color of the background or letters used in DOMESTIC.
2. Retrieve KITE from Activity 4.3 and use it with FRAME created in Reference Point 4 to design your own cross stitch sampler.
3. Design a tessellation to use as a frame for your own cross stitch sampler or the one provided in DOMESTIC.

4. Change the procedure CROSS to a design that resembles a French Knot or some other embroidery stitch.
5. Change the message in the sampler to read "ONE CHAPTER TO GO" or some other appropriate statement.

Recruiting the Turtle's Team

5

Activity 5.1 Sprinting Sprites



Turtle Trials. This activity introduces sprites and how they are controlled with turtle graphics commands.

NOTE Commodore Logo has the capability of manipulating eight separate moveable graphic objects called sprites. The turtle used in Chapters 1 through 4 is sprite 0. The Utilities Disk provides predefined sprites and procedures for your use.

1. Boot the system with Logo. Insert the Utilities Disk in the drive and enter the following commands. Record the results where indicated.

```
READ "SPRITES
DRAW ______________________________________
FD 100 ____________________________________
CS ________________________________________
TELL 1 ____________________________________
ST ________________________________________
RT 45 FD 100 ______________________________
TELL 2 ST
RT 90 FD 100 ______________________________
TELL 6 ST LT 30 BK 50 _____________________
PU TELL 0 HOME ____________________________
TELL 6 HOME _______________________________
```

2. Type the following commands and write your observations on the lines provided.

```
READ "ANIMALS
TELL 0  ____________________________________
GOODBYE
READ "SPRITES
PU TELL 0 RT 45 FD 100  ___________________
TELL 1 ST RT 45 FD 100  ___________________
BK 200  ____________________________________
TELL 2 ST SETH 90 FD 80  __________________
TELL 0 HOME  ______________________________
f5  ________________________________________
f3  ________________________________________
TELL 3 ST  ________________________________
TELL 4 ST SETXY -100 100  _________________
TELL 5 ST LT 225 FD 100  __________________
SETXY XCOR - 40 YCOR + 150  _______________
TELL 6 ST SETY 80  ________________________
TELL 7 ST SETH 90 BK 70  __________________
TELL 8 ST  ________________________________
```

3. Enter the following procedure, then describe the effect it has on the screen display.

```
TO REMOVE.SHAPES :NUMBER
IF :NUMBER = 8 STOP
TELL :NUMBER HOME HT
REMOVE.SHAPES :NUMBER + 1
CS
CLEARTEXT
END

TELL 1 ST BK 50
TELL 2 ST FD 85
TELL 6 ST RT 60 FD 100
TELL 4 ST SETH 289 FD 75
TELL 5 ST SETXY 80 (-65)
REMOVE.SHAPES 0  __________________________
```

4. Retype REMOVE.SHAPES to read as follows:

   ```
   TO REMOVE.SHAPES
   EACH [0 1 2 3 4 5 6 7][HOME HT]
   CS
   CLEARTEXT
   END

   TELL 0
   REMOVE.SHAPES
   ```

   Does the revised version of REMOVE.SHAPES clear the screen as that in Part 3? ______________________________

5. Enter the following three procedures. Then enter the given commands to observe each of the eight sprites within each of the five shape files provided on the Utilities Disk.

   ```
   TO DISPLAY.SHAPES :NUMBER :ANGLE :COLOR
   IF :NUMBER = 8 STOP
   TELL :NUMBER PC :COLOR ST SETH :ANGLE PU
     FD 60
   PR SE [SPRITE NUMBER] :NUMBER
   WAIT 5
   DISPLAY.SHAPES :NUMBER + 1 :ANGLE + 45
     :COLOR
   END

   TO WAIT :TIME
   REPEAT 500*:TIME []
   END

   TO INTRO.SHAPES :NUMBER :ANGLE :COLOR
   DISPLAY.SHAPES :NUMBER :ANGLE :COLOR
   REMOVE.SHAPES
   END

   INTRO.SHAPES 0 0 5   ______________________
   READ "VEHICLES
   INTRO.SHAPES 0 0 0   ______________________
   READ "SHAPES
   ```

```
INTRO.SHAPES 0 0 1 ______________________
READ "ASSORTED
INTRO.SHAPES 0 0 6 ______________________
READ "RUNNER ____________________________
CTRL-G (after a while)
REMOVE.SHAPES
INTRO.SHAPES 0 0 1 ______________________
```

6. Type the following commands and write your observations on the lines provided.

```
READ "ANIMALS
TELL :DINOSAUR ST FD 100 _________________
TELL 0 HT
TELL :BUTTERFLY ST SETH 270 FD 150 _______
TELL :HORSE ST SETXY 120 70 ______________
TELL :BUG ST RT 180 FD 60 ________________
TELL :CAT ST SETX 100 ____________________
TELL :KANGAROO ST ________________________
REMOVE.SHAPES ____________________________
```

7. Enter the following commands and record the results.

```
READ "SPRITEDEMOS
READ "ASSORTED
```

List the names of the procedures in SPRITEDEMOS. ________________

```
TELL 0 HT _______________________________
TELL 4 ST _______________________________
HOP 10 __________________________________
TELL 4 HOME _____________________________
TELL 0 HT _______________________________
HOP 10 __________________________________
PO HOP __________________________________
f1 ______________________________________
f3 ______________________________________
TELL 6 ST _______________________________
CRAWL 5 _________________________________
PO CRAWL ________________________________
```

```
f1  ______________________________________
f3  ______________________________________

REMOVE.SHAPES
```

8. Edit the RACE procedure to read as follows.

```
TO RACE
TELL 0 HT
TELL 4 SETXY -110 30 ST
TELL 6 SETXY -120 (-30) ST
REPEAT 15 [TELL 4 HOP 1 TELL 6 CRAWL 1]
END

RACE  ______________________________________
```

Turtle Talk. Sprites are graphic images that can be moved around the Commodore screen using the various primitives covered in this book. The symbolic turtle you are accustomed to observing is an example of a sprite. The turtle (known as sprite 0) is one of eight sprites numbered 0 through 7 which can be displayed at one time. You can create your own sprites (covered in Activity 5.4) or access any of the five pre-defined shapes files provided on the Utilities Disk. This activity concentrates on displaying and positioning pre-defined shapes.

Part 1 demonstrates that access to sprites is initiated by the command

```
READ "SPRITES
```

The TELL and ST commands must be used together to display a sprite. TELL N, where N is a number from 0 through 7, designates which sprite is to be displayed. ST causes the selected sprite to appear on the screen. The combination of these two commands causes a sprite to appear in the home position on the screen. DRAW or, equivalently, TELL 0 displays the turtle. TELL 1 and TELL 6 in Part 1 illustrate the appearance of undefined sprites. This situation is improved in Part 2 when the ANIMALS shape file is accessed. ANIMALS is one of the five predefined shape files available to you on the Utilities Disk. Table 5-1 defines the shape file names and lists the eight sprites by name and corresponding number for each file.

Table 5-1. Sprite Shapes and Numbers on Utility Disk

#	ANIMALS	#	ASSORTED
1	Dinosaur	1	Target
2	Kangaroo	2	Truck
3	Bug	3	Rocket
4	Dolphin	4	Balloon
5	Horse	5	Bow
6	Cat	6	Arrow
7	Butterfly	7	Man

#	SHAPES	#	VEHICLES
1	Frame	1	Truck
2	Big Ball	2	Car
3	Small Ball	3	Bicycle
4	Square	4	Submarine
5	Triangle	5	Airplane
6	Heart	6	Boat
7	Box	7	Balloon

Parts 1 and 2 demonstrate that sprites can be moved to any screen location by using a variety of direction and movement commands. Once a sprite is displayed it can be used again by using TELL, but ST is not necessary. Notice that sprites that cross the paths of other sprites are not damaged. Sprites have the capability of passing in front of and behind other sprites, giving a three-dimensional effect. This topic will be covered in a future activity.

After sprites are displayed on the screen, DRAW does not remove them as in previous activities. In Part 2, the sequence of commands GOODBYE and READ "SPRITES, and READ "ANIMALS clears the screen for the display of animal sprites, but in an awkward manner. READ "ANIMALS is not necessary as GOODBYE does not erase the file of sprite. Part 3 introduces a procedure named REMOVE.SHAPES for erasing sprites from the screen. This procedure is modified in Part 5 by using the procedure EACH. EACH becomes available after the SPRITES file is read from the Utilities Disk. EACH lets you assign operations to many sprites. The sprites to be affected by EACH are listed within brackets following the command. The second list within brackets defines what the sprites are to do. Procedures DISPLAY.SHAPES, INTRO.SHAPES and WAIT are added in Part 5 to collectively display and remove all sprites from each of the four predefined shape files illustrated in Table 5-1. To color a sprite, the color must be declared after the sprite is identified by the TELL command, but before it appears on the screen. Otherwise, the sprite appears in the previously declared color before changing to the new color. Also, Part 6 demonstrates that sprites can be displayed by name as well as by number.

Several procedures are available on the Utilities Disk for animating sprites. The use of three procedures from SPRITEDEMOS are presented in Parts 7 and 8. HOP causes a sprite to move from left to right with a hopping (up and down) effect. CRAWL causes the sprite to move from the left to right in small randomly generated steps. The RACE procedure, edited in Part 8, suggests how to control the racing of several sprites across the screen.

Procedure HOP and CRAWL are listed below for you to study.

```
TO HOP :N
RT 90
REPEAT :N [FD 10 LT 90 FD 10 RT 90 FD 3 RT 90 FD
  10 LT 90]
LT 90
END

TO CRAWL :N
RT 90
REPEAT :N [FD 1 FD RANDOM 26]
LT 90
END
```

Turtle Truth. Study the following commands and predict the outcome.

```
REMOVE.SHAPES
READ "VEHICLES
```

Edit the RACE procedure to read as follows.

```
TO GREAT.RACE
TELL 0 RT 90 FD 318 HT
TELL 5 SETXY -130 100 ST
TELL 6 SETXY -130 10 ST
TELL 4 SETXY -130 (-40) ST
```

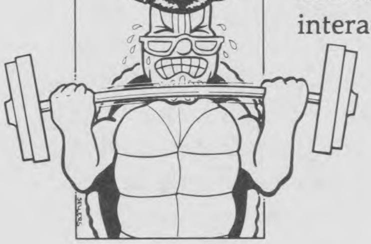
```
REPEAT 10 [TELL 6 HOP 1 TELL 5 CRAWL 3
  TELL 4 CRAWL 2]
END
```

```
GREAT.RACE ______________________________
```

Turtle Tester. Edit the CRAWL and HOP procedures so the vehicles are racing in the appropriate directions. Edit GREAT.RACE so the vehicles start on the right side of the screen. Save the GREAT.RACE by inserting your file disk and saving these procedures under the name COMPETITION. (Answer on page 297.)

Turtle Teaser. Select sprites of your choice and design your own competition.

Activity 5.2 Springing Sprites



Turtle Trials. This activity demonstrates the methods necessary to address specific sprites and control their interaction with the screen display and other sprites.

1. Insert the Utilities Disk and read in the files SPRITES, SPRITEDEMOS, SHAPES, and ARCS. Enter the following commands and describe the results.

```
TELL 0 HT
PU
TELL RANDOM 8 ST ______________________
WHO ___________________________________
TELL RANDOM 8 ST SETXY 50 50 __________
WHO ___________________________________
TELL RANDOM 8 ST SETXY -50 50 _________
WHO ___________________________________
TELL RANDOM 8 ST SETXY -20 (-30) ______
WHO ___________________________________
TELL RANDOM 8 ST SETY 100 _____________
WHO ___________________________________
TELL RANDOM 8 ST SETXY -40 30 _________
WHO ___________________________________
```

2. Enter the following commands and record the results.

```
ASK 0 [PU ST SETXY 70 (-60)] __________
WHO ___________________________________
ASK 0 [PD FD 120 ARCL 60 180 FD 120 LT 90
  FD 120 ] ____________________________
WHO ___________________________________
ASK 0 [PC -1 BK 15 PC 1 LT 90 FD 80] ___
TELL 3 ST SETXY 60 (-60) ______________
CTRL-P (change -60 to 50) _____________
WHO ___________________________________
FD 20 _________________________________
BK 75 _________________________________
FD 60 _________________________________
WHO ___________________________________
ASK 0 [RT 45 FD 21 BK 21 LT 45] _______
WHO ___________________________________
BK 70 _________________________________
FD 70 _________________________________
WHO ___________________________________
```

Does the gate have any effect on stopping the movement of the ball?

____

```
GOODBYE ______________________________
```

3. Enter the following procedure and observe the screen display.

```
TO PINBALL
ASK 0 [PU PC 1 SETXY 70 (-60) PD SETH 0]
ASK 0 [FD 120 ARCL 60 180 FD 120 LT 90 FD
  120 PC -1 BK 15 PC 1 LT 90 FD 80]
ASK 0 [SETH 45 FD 20]
ASK 0 [HT]
END

PINBALL ______________________________
```

4. Add the procedure PREPARE to correct the difficulty encountered in Part 3. Describe the observed display.

```
TO PREPARE
READ "SPRITES
READ "SPRITEDEMOS
READ "SHAPES
READ "ARCS
END

PREPARE ______________________________
PINBALL ______________________________
```

What is the purpose of the procedure PREPARE? ______________
Why isn't it included as part of PINBALL rather than being written as a separate subroutine? ______________________________
Save these procedures as WARM.UPS on your disk.

```
SAVE "WARM.UPS
GOODBYE
```

5. Read the WARM.UPS file from your disk. Then place the Utilities Disk in the drive. Type the following commands and observe the results. What file is listed as not being found? ______________________

```
PREPARE ______________________________
PINBALL ______________________________
TELL 3 ST SETXY 60 (-68) SETH 0 ________
```

Although it is listed as not being found, SHAPES is still in memory. What command indicates this to be true? ____________________

6. Edit PINBALL and enter TILT to read as follows. Observe the interaction of the ball with parts of the pinball display.

```
TO PINBALL
ASK 0 [PU PC 1 SETXY 70 (-60) PD SETH 0]
ASK 0 [FD 120 ARCL 60 180 FD 120 LT 90 FD
  120 PC -1 BK 15 PC 1 LT 90 FD 80]
ASK 0 [PU SETXY -20 80 PD CIRCLER 10]
ASK 0 [PU SETXY 20 80 PD CIRCLER 10]
ASK 0 [PU SETXY 70 60 PD SETH 330 FD 60]
ASK 0 [HT]
END

TO TILT
PINBALL
TELL 3 ST SETXY 60 (-68) SETH 0
FD 300
TRAPLOOP
END

TILT ______________________________
```

Wait until the ball hits some part of the pinball display, then press any key to stop the program.

What happens to the ball when it hits a part of the pinball display?

____

Can the ball escape through the "shoot" if it ricochets in the direction of the "shoot?" ______________________________

7. Let's simulate a gate for the pinball display by adding procedures OPENGATE and CLOSEGATE and by editing TILT, PINBALL, and TRAPLOOP as follows:

```
TO PINBALL
ASK 0 [PU PC 1 SETXY 70 (-60) PD SETH 0]
ASK 0 [FD 120 ARCL 60 180 FD 120 LT 90 FD 120
 PC - 1 BK 15 PC 1 LT 90 FD 80]
ASK 0 [SETH 45 FD 20]
ASK 0 [PU SETXY -20 80 PD CIRCLER 10]
ASK 0 [PU SETXY 20 80 PD CIRCLER 10]
ASK 0 [PU SETXY 70 60 PD SETH 330 FD 60]
ASK 0 [HT]
END

TO TILT
PINBALL
TELL 3 ST SETXY 60 (-68) SETH 0
OPENGATE
FD 300
TRAPLOOP
END

TO TRAPLOOP
IF RC? STOP
IF TB? BK 1 RT RANDOM 180 CLOSEGATE
FD 1
TRAPLOOP
END

TO OPENGATE
ASK 0 [PC -1 PU SETXY 55 20 PD SETH 45 FD 20 PC 1]
END

TO CLOSEGATE
ASK 0 [PU SETXY 55 20 PD SETH 45 FD 20]
END

TILT
```

Observe the pinball display a few moments, then press any key to stop the program.

8. Edit PINBALL to read as follows. Observe the interaction of the ball with parts of the pinball display.

```
TO PINBALL
ASK 0 [PU PC 1 SETXY 70 (-60) PD SETH 0]
ASK 0 [FD 120 ARCL 60 180 FD 120 LT 90 FD 120
 PC - 1 BK 15 PC 1 LT 90 FD 80]
```

```
ASK 0 [SETH 45 FD 20]
TELL 5 ST SETXY -20 80
TELL 1 ST SETXY 20 80
ASK 0 [PU SETXY 70 60 PD SETH 330 FD 60]
ASK 0 [HT]
END

TILT
```

Wait until the ball makes contact with a sprite, then press any key to stop the program.
What happens when the ball makes contact with either the box sprite or shaded triangle sprite in the pinball display? ______________________

9. Edit TRAPLOOP to read as follows. Observe the interaction of the ball with parts of the pinball display.

```
TO TRAPLOOP
IF RC? STOP
IF TB? BK 3 RT RANDOM 180 CLOSEGATE
IF TS? BK 3 LT RANDOM 180
FD 1
TRAPLOOP
END

TILT
```

Wait until the ball makes contact with the pinball display and a sprite, then press any key to stop the program.
What happens when the ball makes contact with either sprite in the pinball display? ______________________

10. Edit TRAPLOOP to read as follows:

```
TO TRAPLOOP
IF RC? STOP
IF TB? BK 3 RT RANDOM 180 CLOSEGATE
IF TS? BK 3 LT RANDOM 180
FD 5
TRAPLOOP
END

TILT
```

Does the speed of the animated ball improve? ____________

What changes eventually occur? (If the ball does not escape the boundaries of the pinball display, let TILT run longer.) ____________

Turtle Talk. The Commodore Logo Utilities Disk contains several files that facilitate the control of sprites and their interaction with the screen display and other sprites. The SPRITES file gets things started. This activity introduces several useful procedures in the SPRITEDEMOS file.

Activity 5.1 demonstrates that sprites are selected by using the TELL command. Part 1 of this activity shows that the WHO command can be used to keep track of the number of the current sprite. The ASK procedure is presented in Part 2. This procedure lets you manipulate shapes without disturbing the current sprite. The procedure is listed below.

```
TO ASK :N :COMMAND.LIST
LOCAL "CURRENT
MAKE "CURRENT WHO
TELL :N RUN :COMMAND.LIST
TELL :CURRENT
END
```

Local variable CURRENT is used to hold the most recently used sprite by receiving the value of WHO. N is the number of the sprite that you wish to give commands. The list of commands the sprite is to perform is included within brackets ([ ]) and stored in the variable COMMAND.LIST.

Part 2 begins the process of creating a simulated pinball machine. A gate is included in the "shoot" of the design, but it does not prevent the passage of the ball. TRAPLOOP is a procedure contained in SPRITEDEMOS, which tests for the interaction of a sprite with a screen design. SPRITEDEMOS also includes ASK. The PINBALL procedure designed in Part 3 cannot be performed until SPRITEDEMOS is read in Part 4. A WARM.UPS file is defined in Part 5 to provide access to the files necessary to display the pinball display. However, the Utilities Disk still must be used to load the SHAPES sprite file.

Parts 6 through 10 add more detail to the pinball display. Parts 6 and 7 insert bumpers that are part of the background screen display. The TB? procedure within TRAPLOOP controls the interaction of sprites (the ball) with the graphics screen (the bumpers). The TB? procedure gives a value of TRUE when a SPRITE touches a dot on the graphics screen that is different from the background color. In this case the sprite moves back a designated number of units and turns right with random values up to 180 degrees.

When sprite shapes replaced graphic designs for bumpers in Part 8, TRAPLOOPS failed to respond to the interaction of the ball sprite with the bumper sprites. Part 9 corrected this situation by adding the TS? procedure to TRAPLOOP. TS? controls the interaction of a sprite with other sprites in a manner similar to TB? controlling the interaction of a sprite with graphic designs.

The termination of the TRAPLOOP procedure is controlled by the RC? procedure. RC? checks for a keyboard press. When a key is pressed the procedure stops.

The speed of the animated ball is accelerated in Part 10, but not without some strange results. The ball eventually escapes from the enclosed region when the FD command within TRAPLOOP has its argument increased beyond 1.

Turtle Truth. PINBALL is redesigned with different bumper parts in color. The procedure REMOVE.SHAPES is added to TILT to erase the screen for successive executions of the pinball design. Enter the new procedure and edit existing ones to read as follows. Figure 5-1 illustrates the finished product.

```
TO PINBALL
ASK 0 [PU PC 1 SETXY 70 (-60) PD SETH 0]
ASK 0 [FD 120 ARCL 60 180 FD 120 LT 90 FD 120
 PC - 1 BK 15 PC 1 LT 90 FD 80]
TELL 2 PC 2 ST SETXY 35 50
TELL 4 PC 7 ST SETXY -20 80
TELL 5 PC 13 ST SETXY -20 (-30)
TELL 6 PC 6 ST SETY 100
TELL 7 PC 4 ST SETXY -10 30
ASK 0 [PU SETXY 70 60 PD SETH 330 FD 60]
ASK 0 [HT]
END

TO TRAPLOOP
IF RC? STOP
IF TB? BK 3 RT RANDOM 180 CLOSEGATE
IF TS? BK 3 LT RANDOM 180
```

```
FD 1
TRAPLOOP
END

TO TILT
REMOVE.SHAPES
PINBALL
TELL 3 PC 1 ST SETXY 60 (-68) SETH 0
OPENGATE
FD 300
TRAPLOOP
END

TO REMOVE.SHAPES
EACH [0 1 2 3 4 5 6 7][HOME HT]
CS
CLEARTEXT
END

TILT
```

After you have observed the pinball display for a while, press any key to stop the program.

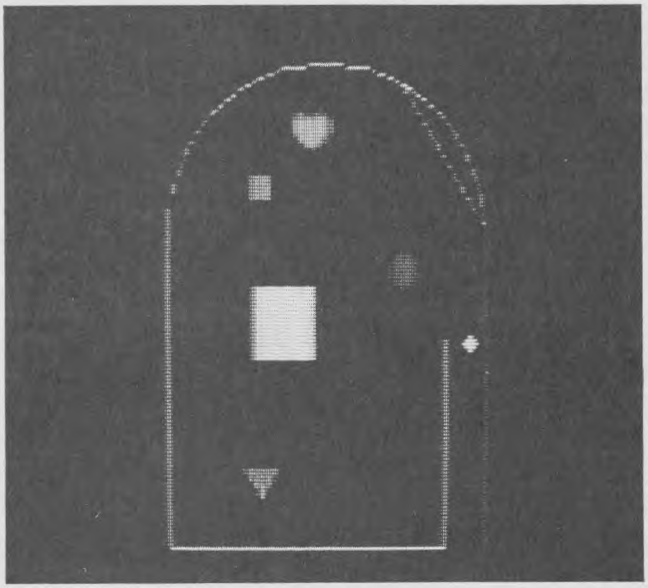
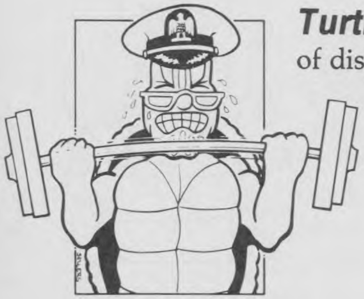


Figure 5-1.

Turtle Tester. Modify the Turtle Truth so that (a) the ball changes color randomly each time it hits a boundary of the pinball design, and (b) the five sprite bumpers flash in random color each time the ball hits them. Save the procedures with the file name GAME1. (Answer on page 297.)

Turtle Teaser. Create a design of your choice that encloses colored sprite shapes. Use TRAPLOOP to animate another sprite to interact with your design. Save the procedures with a name of your choice.

Activity 5.3 Scaling Sprites



Turtle Trials. This activity will introduce the method of displaying sprites in various proportions.

1. Boot the system with Logo. Insert the File Disk and READ in the files SPRITES, SPRITEDEMOS. Enter the following commands and record your observations.

```
READ "VEHICLES
SHOW ____________________________________
READ "ANIMALS
SHOW ____________________________________
READ "SHAPES
```

```
SHOW ______________________________
READ "ASSORTED
SHOW ______________________________
```

2. Enter the following commands and examine the procedures that caused the previous displays. Explain them in your own words.

```
f1
PO SHOW __________________________
PO SHOWLOOP ______________________
f3
```

3. Define procedures HIDE and HIDELOOP by editing SHOW and SHOWLOOP to read as follows.

```
TO HIDE
LOCAL "OLD
MAKE "OLD WHO
TELL 0 HT
HIDELOOP 1 120 100
TELL :OLD
END

TO HIDELOOP :CURRENT :X :Y
IF :CURRENT = 8 STOP
TELL :CURRENT SETSHAPE :CURRENT
SETXY :X :Y
SMALLX SMALLY HT
HIDELOOP :CURRENT + 1 :X :Y - 30
END

SHOW ______________________________
HIDE ______________________________
WHO _______________________________
```

4. Enter the commands and observe the change in appearance. Record your observations.

```
SHOW ______________________________
TELL 2 HOME _______________________
BIGX ______________________________
```

```
SMALLX ______________________________
BIGY ________________________________
SMALLY ______________________________
BIGX BIGY ___________________________
SHOW ________________________________
```

5. Enter the following commands and record your observations.

```
TELL 1 HT
TELL 2 SETSHAPE 5 ___________________
TELL 3 SETSHAPE 5 ___________________
BIGX ________________________________
TELL 5 SETSHAPE 6 ___________________
BIGX ________________________________
BIGY ________________________________
TELL 6 HT ___________________________
TELL 7 SETSHAPE 1 ___________________
BIGX BIGY ___________________________
SHOW
GOODBYE
```

Turtle Talk. Activity 5.2 introduces you to TRAPLOOP, a procedure contained in SPRITEDEMOS and to TB?, TS?, and RC? in SPRITES. This activity continues to explore other built-in procedures within these files.

From SPRITEDEMOS, procedures SHOW and SHOWLOOP display all the sprites within a shape file. Part 1 uses SHOW to display the sprites within VEHICLES, ANIMALS, SHAPES, and ASSORTED. SHOW uses the local variable OLD to remember the current sprite. SHOWLOOP positions the seven sprites vertically in the right portion of the screen. These procedures are listed in Part 2 for you to examine. Part 3 develops HIDE and HIDELOOP to remove the sprites from the screen in a manner similar to the way they are displayed.

The SPRITES file contains four commands that alter the size of sprites. Part 5 demonstrates that BIGX doubles the width of a sprite and BIGY doubles its height. The combination, BIGX BIGY, doubles the width and height of the sprite. SMALLX and SMALLY return the sprites to their original condition. The SHOW procedure also returns all sprites to their original shapes.

The SPRITES file also contains the SETSHAPE command. SETSHAPE becomes useful when you want the same sprite to appear more than once on the same screen display. The command TELL 3 SETSHAPE 5 defines sprite number 3 to be the same sprite shape as sprite number 5. Again, SHOW restores the sprites to their original shapes.

Turtle Truth. The TARGET.PRACTICE procedure listed below suggests the steps to follow when creating new ideas for graphics displays involving sprites. DECIDE first displays all the sprites within a shape file using SHOW. Then various proportions of certain sprites are drawn to determine which sizes are most appropriate for the idea. SETUP performs the actual drawing of the idea, while REMOVE eventually clears the screen. Enter the following procedures and observe the screen display. Carefully study what contribution each command makes to the final drawing illustrated in Figure 5-2.

```
READ "SPRITES from the Utilities Disk
READ "SPRITESDEMOS from the Utilities Disk
READ "STAMPER from the Utilities Disk
READ "ASSORTED from the Utilities Disk
READ "WARM-UPS from the Utilities Disk

TO TARGET.PRACTICE
FULLSCREEN
DECIDE
WAIT 5
SETUP
WAIT 8
REMOVE
SPLITSCREEN
END

TO DECIDE
SHOW WAIT 10
TELL 1 SETSHAPE 6
TELL 2 SETSHAPE 6
BIGX
TELL 3 SETSHAPE 6
BIGX BIGY
TELL 4 HT
TELL 5 BIGX BIGY
TELL 6 HT
TELL 7 SETSHAPE 1
BIGX BIGY
END

TO WAIT :TIME
REPEAT 300*:TIME[]
END

TO SETUP
EACH [0 2 3 4 6][HT]
TELL 7 PC 7 SETXY 125 30
BIGX BIGY
TELL 5 PC 9 SETXY -150 35
BIGX BIGY
TELL 1 PC 5 SETXY -135 20
BIGX
WAIT 5
TELL 5 SMALLX
TELL 1 SHOOT 118
TELL 0 PU SETXY -20 (-40) STAMP [BULL'S EYE]
END

TO SHOOT :MANY
SETH 90
REPEAT :MANY [FD 2]
END

TO REMOVE
EACH [0 1 5 7][HOME HT]
CS
END

TARGET.PRACTICE
GOODBYE
```

Turtle Tester. Read in the VEHICLES file and create a procedure that displays an airplane moving at a steady speed across a screen while a balloon drops down and enlarges as it gets closer to the ground. (Answer on page 299.)

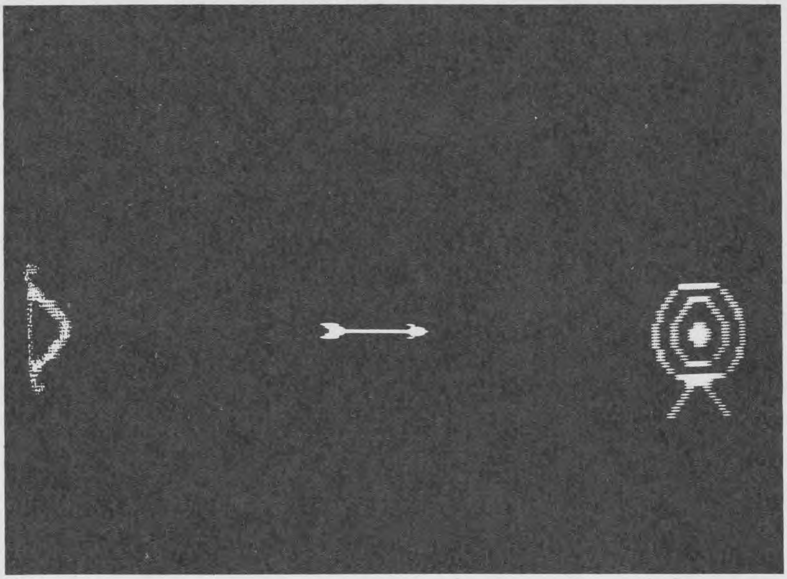
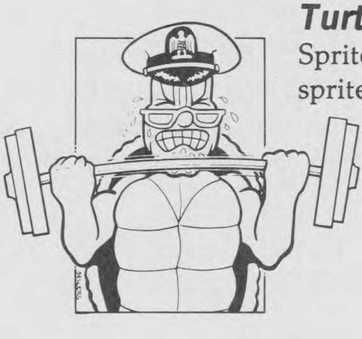


Figure 5-2.

Turtle Teaser. Draw a background that resembles outerspace. Have a rocket blast off decreasing in size as it ascends.

Activity 5.4 Shaping Sprites



Turtle Trials. This activity permits the use of the Sprite Editor in the file SPRED to modify or create sprites.

1. After booting the system with Logo and inserting the Utilities Disk, enter the following commands.

```
READ "VEHICLES
READ "SPRED
TELL 1 ST
EDSH
```

NOTE The following sections illustrate a drawing format that will let you alter a current sprite. The block-like grid on the left of the screen simulates a drawing tool that enables you to design a sprite. Figure 5-3 illustrates the dimensions of the sprite grid on the screen. Each block represents a pixel of the sprite. A block or pixel may be shaded or unshaded. The keystrokes presented in Parts 2 through 7 will demonstrate how to shade or unshade the grid.

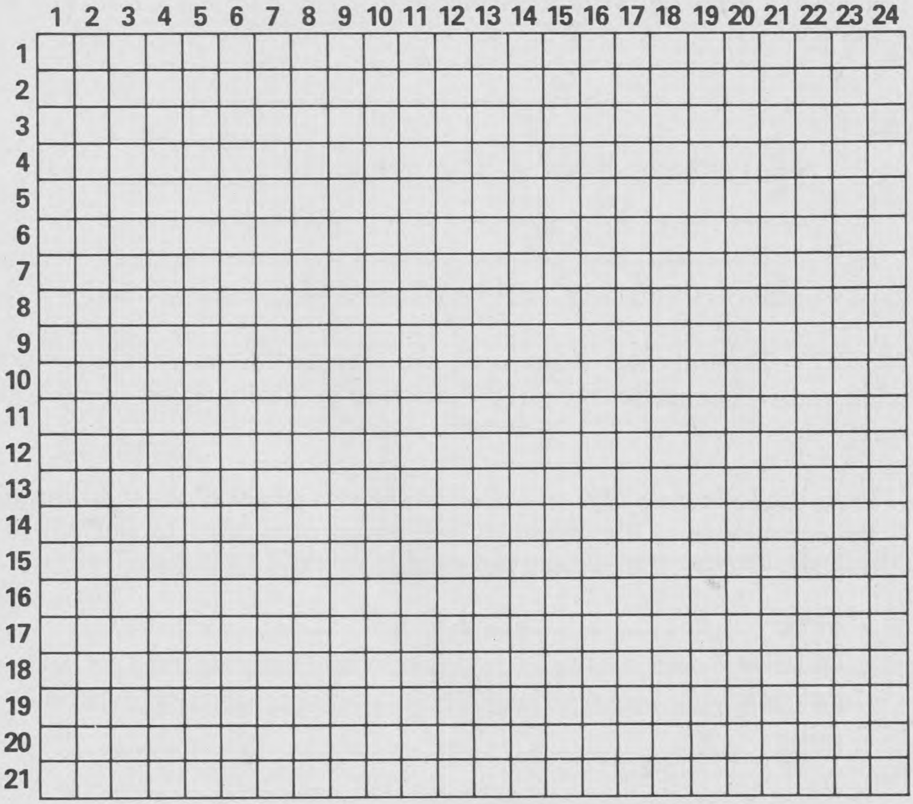


Figure 5-3.

2. Execute the following keystrokes and note the results. The responses may be quite slow, especially with automatic repeats and successive entries. Be patient! The instruction → means to press the CRSR key with the →. ← means to press the SHIFT key simultaneously with the CRSR key with the →.

→ ______________________________
→ ______________________________
→ (hold key down a few seconds) ________
← ______________________________
← ______________________________
← (hold key down a few seconds) ________

NOTE The indicator remains on the horizontal line on which it is positioned and continuously wraps the screen unless commanded otherwise.

3. Continue entering the commands and observe the vertical movement of the indicator. The ↓ means to press the CRSR key with the ↓. The ↑ means to press the SHIFT key simultaneously with the CRSR key and the ↑.

↑ ______________________________
↑ ______________________________
↑ (hold keys down a few seconds) ________
↓ ______________________________
↓ ______________________________
↓ (hold down key a few seconds) ________

NOTE The indicator remains on the vertical line on which it is positioned and will continuously wrap the screen vertically unless commanded otherwise.

4. Press the following keys and observe the movement of the indicator and the appearance of the sprite. HOME means press the CLR/HOME key. Similarly, RETURN means press the key marked RETURN.

HOME ______________________________
RETURN ______________________________
RETURN ______________________________
HOME ______________________________
X ______________________________
X ______________________________
Y ______________________________

`Y` ____________________
`X` ____________________
`Y` ____________________
`XY` ____________________

NOTE The toggling of the X and/or Y keys allows you to preview the sprite in each of its four sizes.

`CTRL 9` ____________________
`CTRL 9` ____________________

5. Observe the steps used to shade or unshade pixels. The instructions that appear in compressed print are key presses rather than characters to be typed. CLR means to press the SHIFT key and the CLR/HOME simultaneously.

`+` ____________________
`→` ____________________
`+` ____________________
`→` ____________________
`*` ____________________
`*` ____________________
`*` ____________________
`*` ____________________
`*` ____________________
`HOME` ____________________
`→` ____________________
`→` ____________________
`SPACEBAR` ____________________
`SPACEBAR` ____________________
`CLR` ____________________

6. Study Figure 5-4. Use the editing commands to create the sprite illustrated.

7. Use X, Y toggles to observe the various shapes the sprite can assume. Describe the changes in the size of the sprite on the indicated lines.

`X` ____________________
`Y` ____________________
`Y` ____________________
`X` ____________________

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1																								
2																								
3																								
4																								
5																								
6		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						
7		●										●							●					
8		●										●	●	●	●					●				
9		●										●			●						●			
10		●										●			●							●		
11		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
12		●										●											●	
13		●				●	●					●							●	●			●	
14		●			●	●	●	●				●						●	●	●	●		●	
15	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
16						●	●												●	●				
17																								
18																								
19																								
20																								
21																								

Figure 5-4.

8. Insert your file disk and Utilities Disk when requested. Follow the directions below to save a newly created sprite and manipulate the shape files.

```
RUN/STOP ____________________________________
```

Insert your file disk.

```
SAVESHAPES "TRAVEL
FD 50 ______________________________________
SETXY 100 100 ______________________________
LT 90 FD 100 _______________________________
WHO ________________________________________
TELL 1 HT __________________________________
TELL 3 ST FD 50 ____________________________
TELL 1 ST  ________________________________
GOODBYE
```

Insert the Utilities Disk.

```
READ "SPRED
READ "VEHICLES
TELL 0 HT
TELL 1 ST
```

Insert your file disk.

```
READSHAPES "TRAVEL
ASK 0 [PU FD 50 ST PD] ___________________
RT 90 FD 50  _____________________________
GOODBYE
```

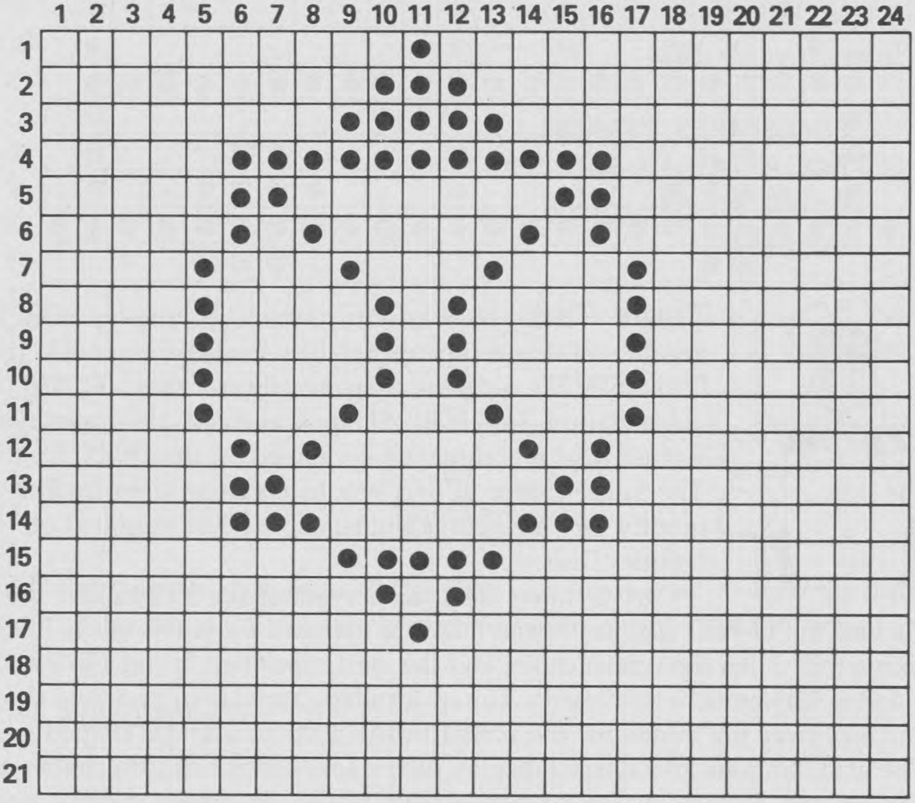
Turtle Talk. This activity introduces you to SPRED, another of the many useful files found on the Utilities Disk. SPRED contains a sprite editor named EDSH (an abbreviation for EDit SHape) and all the previously learned sprite commands contained in the SPRITES file. The Sprite Editor allows you to create new sprite files or modify any of the five built-in sprite files supplied on the Utilities Disk.

Part 1 shows that, after reading the VEHICLES shape file and the SPRED file, the Sprite Editor is accessed by entering the EDSH command. Then the screen clears and the sprite specified in the TELL command is displayed in two forms. The sprite shape appears in its actual shape and size near the middle of the screen to the right of a block shaped grid. The grid contains an enlarged display of the sprite shape. Sprite shapes are constructed by turning on or turning off pixels (or screen display dots) within the grid. Figure 5-3 illustrates that the grid is formed by 21 rows and 24 columns, giving a total of 504 pixels.

The sprite editor is used to move the cursor within the grid. Parts 2 through 7 let you experience the various cursor keys used for positioning the cursor on a pixel, turning on or turning off the pixel, and examining expanded sizes of the shape sprite being constructed. A summary of these cursor keys occurs in Reference Point 5.

The modified shape file is saved on disk in Part 8 with the SAVESHAPES procedure. This file can be retrieved from disk by reading READSHAPES followed by the shape file name.

Turtle Truth. Study the illustration in Figure 5-5 and create a new sprite using +, -, *, SPACEBAR and other editing functions.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1											●													
2										●	●	●												
3									●	●	●	●	●											
4						●	●	●	●	●	●	●	●	●	●	●								
5						●	●								●	●								
6						●		●						●		●								
7					●				●				●				●							
8					●					●		●					●							
9					●					●		●					●							
10					●					●		●					●							
11					●				●				●				●							
12						●		●						●		●								
13						●	●								●	●								
14						●	●	●						●	●	●								
15									●	●	●	●	●											
16										●		●												
17											●													
18																								
19																								
20																								
21																								

Figure 5-5.

Insert the Utilities Disk.

```
READ "SPRED
READ "ANIMALS
TELL 0 HT
TELL 3 ST
EDSH
CLR
```

Use the sprite editor to create the newly designed turtle. Then toggle with X and Y as indicated below.

```
Y
X
Y
X
CTRL 9
CTRL 9
```

Insert your file disk.

```
SAVESHAPES "CREATURES
GOODBYE
```

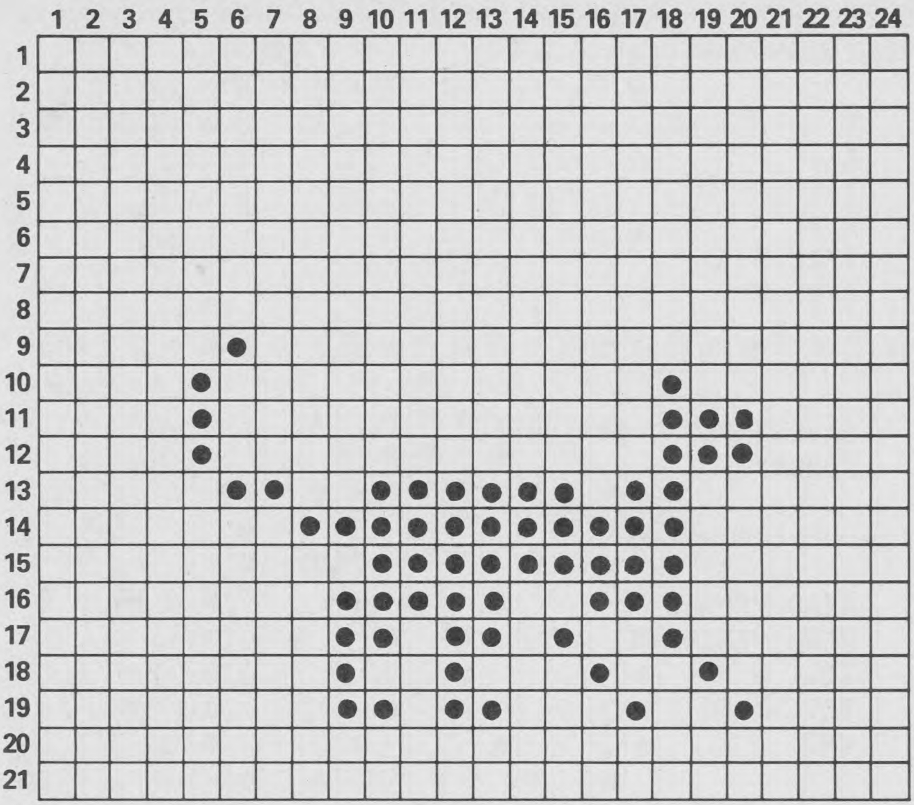
Insert the Utilities Disk.

```
READ "SPRED
READ "ANIMALS
TELL 0 HT
TELL 3 ST
WHO
```

Insert your file disk.

```
READSHAPES "CREATURES
WHO
```

Turtle Tester. Hide the newly created turtle sprite and show sprite #6 (the cat) in this sprite file. Alter the cat to resemble a dog. Use the illustration in Figure 5-6 to help you design its appearance. Save this new sprite in the file CREATURES.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8																								
9						●																		
10					●													●						
11					●													●	●	●				
12					●													●	●	●				
13						●	●			●	●	●	●	●	●		●	●						
14								●	●	●	●	●	●	●	●	●	●	●						
15										●	●	●	●	●	●	●	●	●						
16									●	●	●	●	●			●	●	●						
17									●	●		●	●		●			●						
18									●			●				●			●					
19									●	●		●	●				●			●				
20																								
21																								

Figure 5-6.

Turtle Teaser. Create your own new creature sprite and replace the existing sprite 1, 4, 5, or 7. Save your new sprite with the CREATURES file. You may use Figure 5-3 to design your sprite.

Reference Point 5

Command Review. The following charts contain a summary of all the commands, definitions, and Utility Disk files introduced in this chapter.

Table 5-2. Sprite Editor

Key	Result
	Movement Commands
→	Moves indicator one space to the right
←	Moves indicator one space to the left
↓	Moves indicator one space down
↑	Moves indicator one space up
RETURN	Moves indicator to beginning of next line
HOME	Moves indicator to upper left hand corner of the grid
	Drawing Commands
+	Shades pixel or box under indicator
*	Shades pixel or box under indicator and moves the indicator one space to the right
—	Removes shading of pixel or box under indicator
SPACE BAR	Removes shading of pixel or box under indicator and moves the indicator one space to the right
INST DEL	Moves indicator one space left and erases pixel or box
CLR	Erases all pixels or boxes and places indicator in upper left hand corner
	Screen Display Commands
X	Toggles BIGX/SMALLX display
Y	Toggles BIGY/SMALLY display
CTRL 9	Toggles reverse display of sprite
	Defining a Sprite
CTRL-C	Assigns existing grid display to sprite shape
RUN STOP	Same as CTRL-C

Table 5-3. Files on Utilities Disk

SPRITES
TS?
EACH
EACH1
ASK
SMALLX
BIGX
SMALLY
BIGY
EXP2
TB?

SPRED
ED.LOOP
REDISPLAY
ED.COMMAND
RC&FLASH
ED.FRAME
DECY
DECX
INCY
INCX
SETPOINT
ERASEPOINT
CLEARSHAPE
TOGGLEX
RVS.SHAPE

SPRED (con't.)
ED.QUIT
TOGGLEPOINT
FLASH&WAIT
WHOSHP
SP.ADDR
SP.BIT
BITON
SETUP
SAVESHAPES
READSHAPES
EDSH

SPRITEDEMOS
ASK
CRAWL
HOP
RACE
SHOW
SHOWLOOP
TRAP
BOX
TRAPLOOP

STAMPER
STAMP
STAMP1

Application. Enter the following procedures and define the new sprite illustrated in Figure 5-7. Run RACE to recreate the story of the tortoise and the hare. Notice the priority system as one sprite passes over the other. SAVE the procedures by the name TORT.&.HARE.

```
Boot the system with Logo
Insert the Utilities Disk
READ "STAMPER
READ "SPRITEDEMOS
READ "SPRED
READSHAPES "CREATURES

EACH [0 1 2 3 4 5 6 7][HT]
TELL 2 ST
EDSH
```

Change Sprite #2 (kangaroo) to resemble the hare in Figure 5-7 and define it with RUN/STOP key.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1													●	●				●	●					
2												●		●	●		●	●						
3												●			●	●	●							
4														●	●	●		●						
5														●	●	●	●	●	●					
6															●	●	●	●						
7														●	●	●	●							
8													●	●	●	●	●			●				
9												●	●	●	●	●	●	●		●	●			
10								●	●			●	●	●	●	●	●	●	●	●				
11							●			●	●	●	●	●	●	●		●	●	●				
12							●			●	●	●	●	●	●	●								
13								●	●		●	●	●	●	●	●								
14												●	●	●	●	●	●							
15													●	●	●	●	●							
16														●	●	●								
17													●	●	●									
18												●	●											
19												●	●	●	●	●	●							
20																								
21																								

Figure 5-7.

```
TELL 2 HT
TELL 1 ST
EDSH
```

Change Sprite #1 (dinosaur) to resemble the boulder in Figure 5-8 and define it with RUN/STOP key.

Change Sprite #4 (dolphin) to resemble the tree in Figure 5-9 and define it with the RUN/STOP key.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1																								
2																								
3																								
4																								
5																								
6																								
7																								
8								●	●	●														
9								●	●	●					●	●								
10							●	●	●	●	●			●	●	●	●	●						
11							●	●	●	●	●	●		●	●	●	●	●	●					
12						●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				
13						●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●			
14					●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
15					●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
16				●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
17				●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
18		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
19		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
20	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
21	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Figure 5-8.

Insert your file disk

```
SAVESHAPES " FABLE.CHAR.
```

Edit the HOP and CRAWL procedures to read as follows:

```
TO HOP :N
REPEAT :N [FD 3 LT 90 FD 3 RT 90 FD 3 RT 90 FD 3
  LT 90]
END

TO CRAWL :N
REPEAT :N [FD 3 WAIT 3 FD 2]
END
```

Enter the following procedures. Try to follow the logic and predict the final screen display.

```
TO TREES
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1									●	●	●	●	●	●										
2				●	●		●	●						●	●	●	●	●	●					
3			●			●														●				
4			●																		●	●		
5			●	●	●																	●		
6					●																	●		
7					●	●	●													●	●	●		
8							●											●	●					
9								●	●					●		●	●							
10										●	●	●	●	●	●									
11											●	●	●	●										
12											●	●	●	●										
13											●	●	●	●										
14											●	●	●	●										
15											●	●	●	●										
16											●	●	●	●										
17											●	●	●	●										
18											●	●	●	●										
19										●	●	●	●	●	●									
20							●	●	●			●	●		●	●	●							
21						●	●					●					●	●	●					

Figure 5-9.

```
TELL 4 PC 8 SETXY -130 10 ST
EACH [5 6 7][SETSHAPE 4]
TELL 5 SMALLX BIGY SETXY -10 10 PC 9 ST
TELL 6 BIGX BIGY SETXY 50 60 PC 11 ST
TELL 7 BIGX SMALLY SETXY -10 110 PC 10 ST
END

TO COURSE
PC 4
SETXY -140 (-60)
PD
FD 40 RT 45 FD 100 RT 60
FD 50 LT 45 FD 60 LT 40
FD 30 SETH 90 FD 112
PU SETXY -55 (-60) SETH 0
PD FD 30 RT 45 FD 15 RT 60
FD 20 LT 45 FD 190 LT 30
FD 30 HT
TELL 1 PC 12 SETXY -50 (50) ST
TREES
END

TO MARKS
TELL 2 PC 1 ST
SETXY -135 (-60)
TELL 3 PC 5 ST
SETXY -70 (-60)
END

TO PREPARE
FULLSCREEN
BG 13
TELL 0 PC 8 ST PU
SETXY -120 (-60)
STAMP "START
SETXY 50 100
STAMP [FINISH LINE]
END

TO STORY
TELL 2 SETH 35 HOP 22
TELL 3 SETH 0 CRAWL 10
RT 45 CRAWL 8
RT 45 CRAWL 13
LT 90 CRAWL 4
TELL 2 SETH 90 HOP 15
SETH 180 HOP 3
LT 90 HOP 6
TELL 3 SETH 30 CRAWL 19
STAMP [I WIN!!!]
TELL 2 FD 5 STAMP "PHOOEY!!!
END

TO WAIT :TIME
REPEAT :TIME*300[]
END

TO FABLE
READSHAPES "FABLE.CHAR
```

```
PREPARE
COURSE
MARKS
STORY
END

FABLE

SAVE "TORT.&.HARE

GOODBYE
READ "TORT.&.HARE
FABLE
```

Suggested projects

1. Animate another fable with original sprites.
2. Write an original story and animate it with sprites.
3. Create a game incorporating sprites and the TB? and/or TS? procedures.
4. Create an instructional lesson that requires you to move the sprites to the word that names each one, using the STAMP procedure.

Creating Turtle Temptations

6

Introduction

Logo is designed to be a tool for developing thinking skills. This book concentrates on one component of Logo, Turtle Graphics, as a medium by which intellectual growth is stimulated. Through the control of a "turtle," the computer provides an environment in which problem solving processes can occur. Exploration and learning by discovery are necessary ingredients in a Logo learning experience. Chapters 1 through 5 introduced you to the elements of Turtle Graphics with this theme in mind.

You were first asked to explore and play with various Logo commands in the Turtle Trials section. Some results were successful; some were not. You learned correct syntax (language and punctuation) and observed the resulting screen display patterns created by these commands.

Following this experience, you were presented with a Turtle Truth that suggested how these various commands could be combined to display a more sophisticated idea. It is the creating and solving of more sophisticated ideas, based upon the successful resolving of related ideas, that becomes the ultimate goal in the Logo learning situation. Let us explore this concept more fully.

The learner has an idea (or graphic design) in mind. Once having established a problem to be solved, the idea is explored by instructing the computerized turtle to draw a picture on the screen. The resulting image either satisfies the learner or does not. Choices, decisions, and additional experimenting to find an acceptable solution are the steps in this problem solving process.

Once an acceptable solution has been found, the sequence of commands is grouped together to form a procedure and is given a name. This process continues, and the learner develops a collection of procedures that can be used at any time in a new learning sequence. Learning becomes a gradual process of trial and error experiences, where new hypotheses are formed and tested until new resolutions are found.

This individually created language provides an opportunity for unlimited creativity and personal mental activity and development. To nurture this development, learners should choose, design, and implement projects.

Two different approaches can be used to develop these projects: Part-to-Whole Programming and Whole-to-Part Programming. The purpose of Chapter 6 is to present examples of these methods and to provide a list of suggested projects to attempt.

Individual learning styles will influence the approach that the reader selects to use when developing an original project. Both Whole-to-Part and Part-to-Whole methods are equally satisfactory.

Part-to-Whole Programming

Part-to-Whole Programming is an approach where the learner does not have a clearly defined image or design in mind when beginning a project. The programmer begins with a basic idea which, upon its successful completion, suggests an elaboration or additional idea. This process continues until no new ideas are conceived and the final project is completed. The following example demonstrates this problem solving approach.

Let us assume we wish to draw a picture of a computer terminal. Initially, our idea coincides with the diagram illustrated in Figure 6-1. Enter the procedure COMPUTER to obtain the diagram.

```
TO COMPUTER
PU
SETXY -80 (-60)
PD
SETXY -50 (-30)
SETXY -50 70
SETXY -30 80
SETXY 70 80
SETXY 50 70
SETXY -50 70
SETXY 50 70
SETXY 50 (-30)
SETXY -50 (-30)
SETXY 50 (-30)
SETXY 20 (-60)
SETXY -80 (-60)
SETXY -80 (-70)
SETXY 20 (-70)
SETXY 70 (-40)
SETXY 70 80
PU
SETXY -40 (-20)
PD
SETXY -40 60
SETXY 40 60
SETXY 40 (-20)
SETXY -40 (-20)
END
```

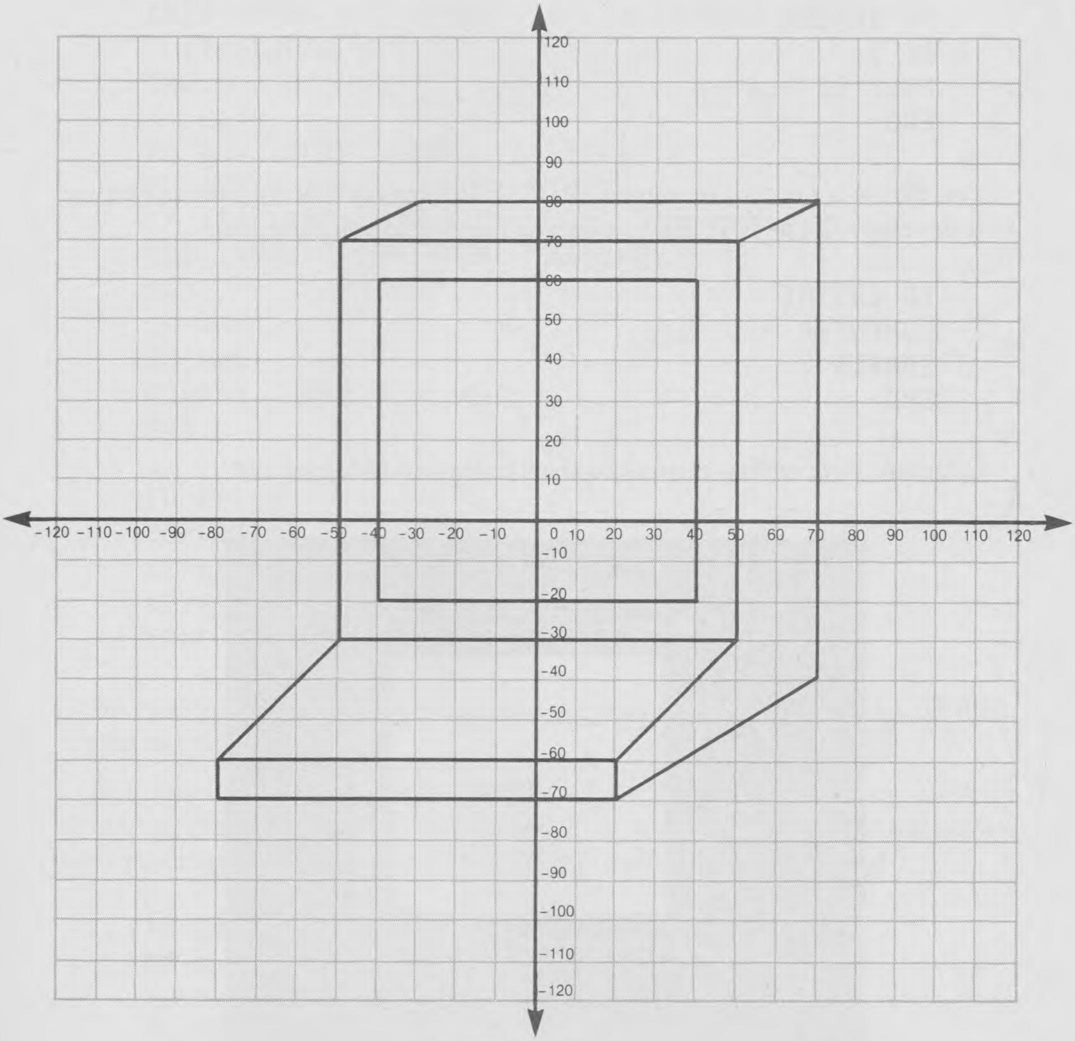


Figure 6-1.

As you observe the computer, you decide its appearance could be improved if the screen were shaded in green. You recall the FILL procedure developed in Activity 3.7 and write a procedure called SCREEN that uses the procedure called FILL. Enter these procedures.

```
TO FILL :SIDE :STEP :MANY
IF :SIDE < :STEP STOP
REPEAT :MANY [FD :SIDE RT 360/:MANY]
FILL :SIDE - :STEP :STEP :MANY
END

TO SCREEN
PC 5
FILL 80 1 4
END
```

You define a procedure named ENTIRE to observe the composite picture as it develops. Type ENTIRE to observe the developing diagram.

```
TO ENTIRE
COMPUTER
SCREEN
END
```

Compare your screen display to the illustration in Figure 6-2.

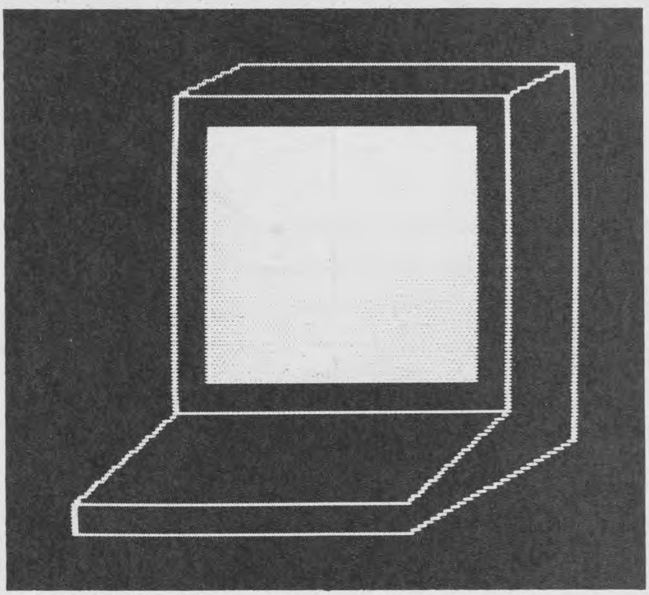


Figure 6-2.

Now that you are satisfied with this diagram, you notice the keyboard portion of the computer is missing keys. You proceed to display keys by preparing a procedure called KEYS, a ROW procedure to be used in keys, a DOT procedure to be used in ROW, and then editing ENTIRE to include the procedures.

```
TO KEYS :XPOS :YPOS :TALLYKEY :TALLYROW
PC 3
ROW :XPOS :YPOS :TALLYKEY
IF :TALLYROW = 4 STOP
KEYS :XPOS - 5 :YPOS - 5 :TALLYKEY
  :TALLYROW + 1
END

TO ROW :XPOS :YPOS :TALLYKEY
PU SETXY :XPOS :YPOS PD DOT
IF :TALLYKEY = 9 STOP
ROW :XPOS + 10 :YPOS :TALLYKEY + 1
END

TO DOT
REPEAT 4 [FD 1 RT 90]
END

TO ENTIRE
COMPUTER
SCREEN
KEYS -50 (-40) 1 1
END
```

By typing ENTIRE again, your new diagram should agree with that illustrated in Figure 6-3.

Still not satisfied with the complexity of your creation, you cleverly decide to create a maze on the screen. This is accomplished by adding the next procedure and modifying ENTIRE to read as follows:

```
TO MAZE :LENGTH
PC 0
FD :LENGTH
RT 90
IF :LENGTH < 0 STOP
MAZE :LENGTH - 5
END
```

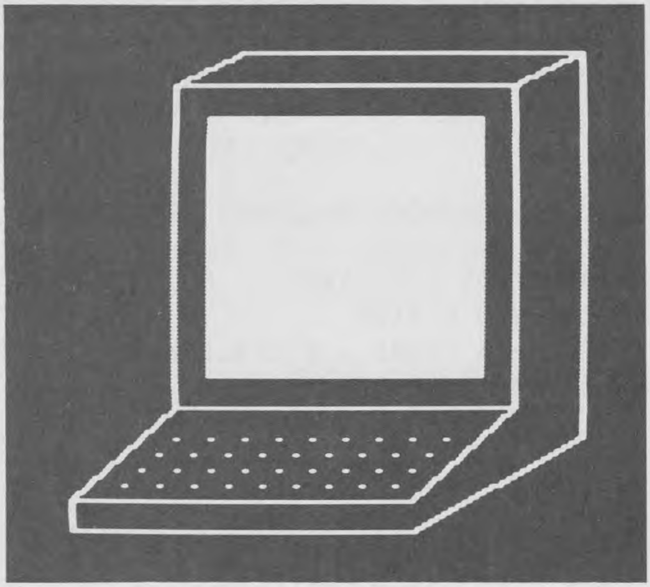


Figure 6-3.

```
TO ENTIRE
HT
COMPUTER
SCREEN
KEYS -50 (-40) 1 1
PU SETXY -30 (-10) PD
MAZE 60
END
```

Figure 6-4 illustrates the new composite diagram that has been created. Type ENTIRE to observe the diagram on your screen display.

Your creativity is really coming alive! You decide to move a sprite turtle through the maze, as if a racing game were being played. The four directions (up, down, left, and right) in the maze normally require four different turtle sprites, one for each of the four directions. Since sprite 0 (the symbolic turtle) is the only sprite shape which actually changes direction, use this for simplicity. This new idea is accomplished by adding MAZE2 and WAIT, and again modifying ENTIRE.

```
TO MAZE2 :LENGTH
PC 1
FD :LENGTH
```

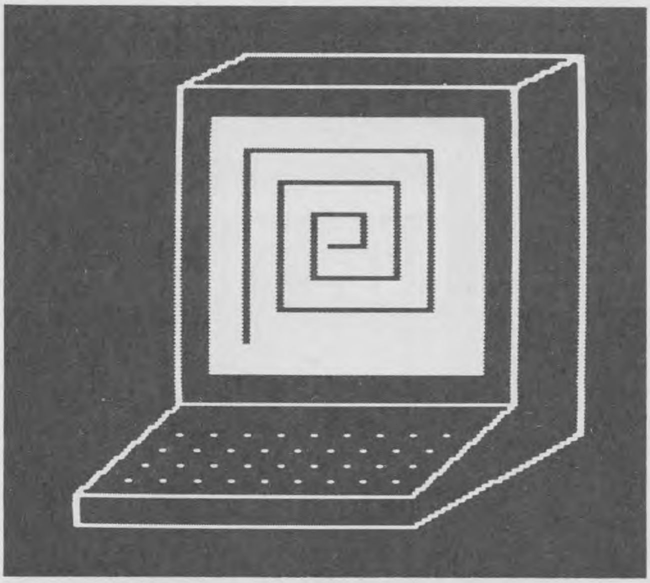


Figure 6-4.

```
WAIT 3
RT 90
WAIT 3
IF :LENGTH < 0 STOP
MAZE2 :LENGTH - 5
END

TO WAIT :TIME
REPEAT 300*:TIME []
END

TO ENTIRE
DOUBLECOLOR
HT
COMPUTER
SCREEN
KEYS -50 (-40) 1 1
PU SETXY -30 (-10) PD
MAZE 60
ST
PU SETXY -25 (-5) SETH 0
MAZE2 50
PD
END
```

By typing ENTIRE, you will observe the diagram illustrated in Figure 6-5.

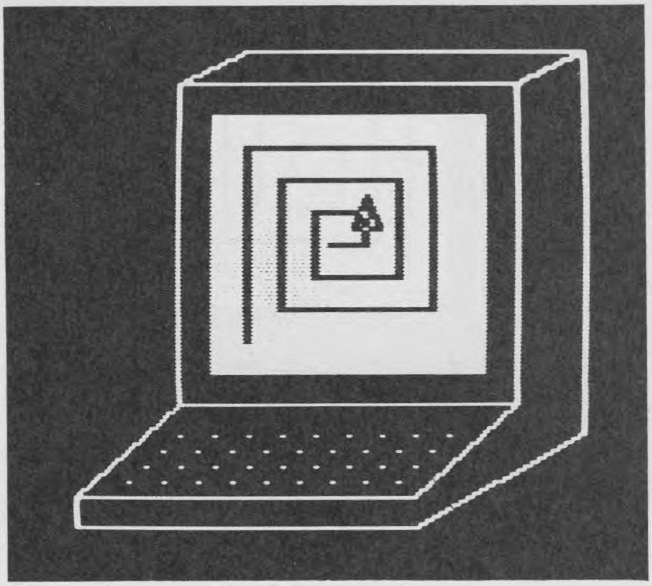


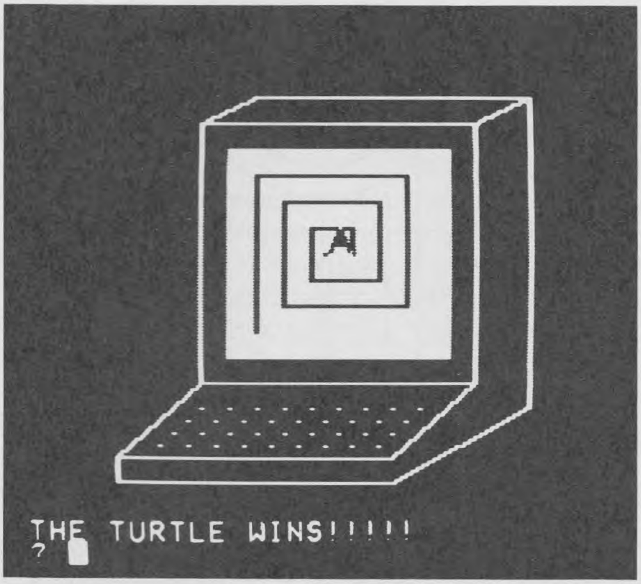
Figure 6-5.

Finally, you decide a message indicating the race has ended is necessary. You also want to give your final idea a more meaningful name, GAME (do this by changing the name of ENTIRE in edit mode). You make the following modifications.

```
TO WINNER
SETH 0
SETXY 0 20
PR [THE TURTLE WINS!!!!!]
REPEAT 10 [HT WAIT 2 ST]
END

TO GAME
DOUBLECOLOR
HT
COMPUTER
SCREEN
KEYS -50 (-40) 1 1
PU SETXY -30 (-10) PD
MAZE 60
ST
PU SETXY -25 (-5) SETH 0
```

```
MAZE2 50
WINNER
END
```

Type GAME to observe the final diagram; Figure 6-6 illustrates how it should appear. Also save your proudly created project as VIDEO.GAME. Remember to GOODBYE before beginning the next section.



```
THE TURTLE WINS!!!!!!
?
```

Figure 6-6.

Whole-to-Part Programming

Whole-to-Part Programming is an approach where the learner has a clearly defined idea or design in mind. The entire drawing is observed, studied, then divided into smaller parts. Each part is given a name, for which a procedure will be developed. Procedures are developed for each of the defined subparts. These procedures can be executed as they are developed (to see the project grow) or all at one time (to see the project at once).

Often the ambitious programmer may conceive an idea with subparts that must be modified, eliminated, or substituted from the original plan. Occasionally, you will find that a mistake in a procedure creates a more pleasing display than the one originally intended, so it is incorporated into the final project. The following steps lead to successful Whole-to-Part Programming.

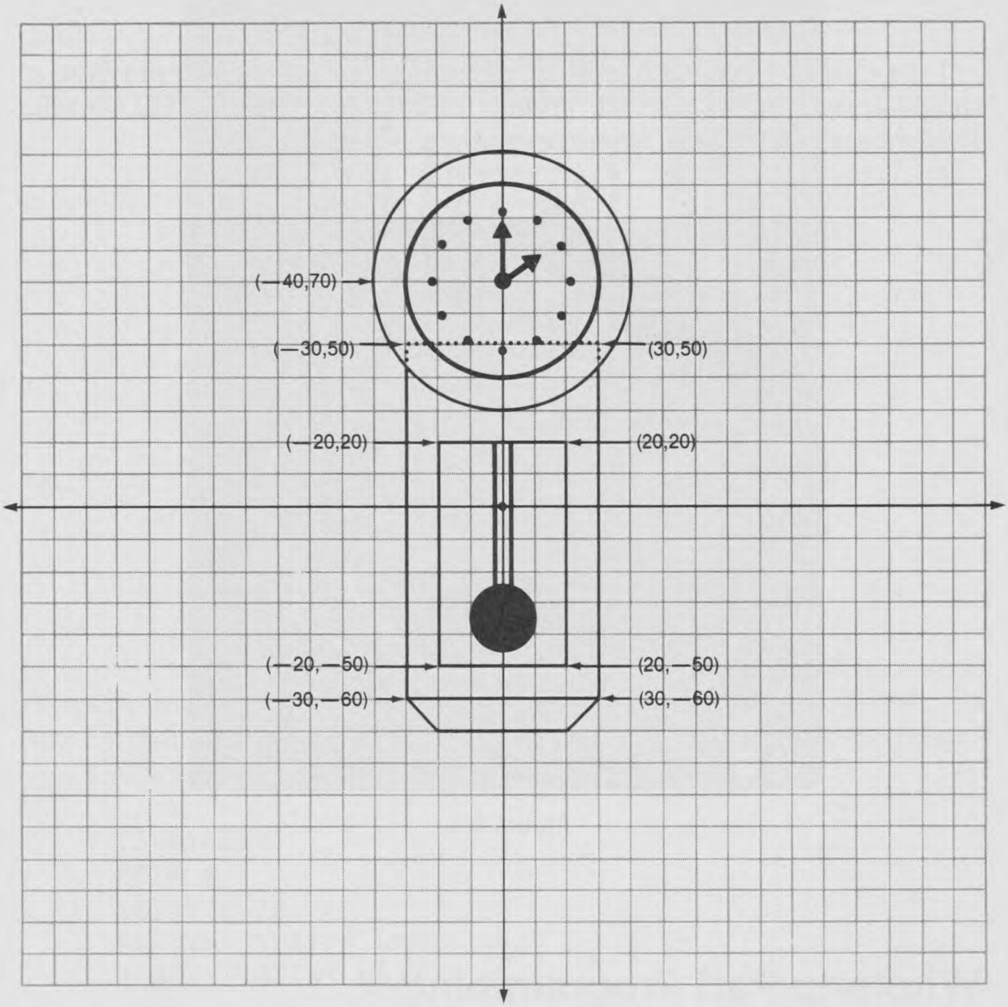


Figure 6-7.

Step 1. The first step in the Whole-to-Part approach to programming is to study the entire picture in detail. Examine the picture for structures that are similar and can be designed as general purpose procedures, where different input variables distinguish one application of a procedure from another. Studying the picture in Figure 6-7, notice there are two rectangles: the outer middle case and the window for viewing the pendulum. There are also two circles that form the upper clock case frame. The bottom of the case is a unique trapezoidal shape. The hands, initials, and dots for numbers on the clock are also distinct, non-repetitive drawings.

Step 2. The next step is to note any special features regarding the diagram. For example, let's suppose the clock is hanging on a wall that is wallpapered. In order to avoid the transparent interference of the clock and the wallpaper, the clock should be shaded rather than outlined.

Let's agree that the main clock case (the circular top, middle rectangle, and lower trapezoid) will be orange. The viewing window and dial (circular area where numbers and hands appear) will be white. The pendulum, dots for numbers, initials, and hands will be black. Some highlighting in black will also give depth to the diagram.

Step 3. The third step is to determine a sequence by which the diagram will develop. In some cases this step is not important; in others it is.

In our example, the background wall would be displayed first. Then the clock will be drawn on top with the shading covering the wall appearing in those locations. Also, the longer rectangle needs to appear as if it is behind the circular part of the clock frame and dial, as indicated by the dotted line in Figure 6-7. In addition, it is convenient to shade the viewing window over the large rectangular part of the frame. The dial must be shaded white prior to the display of dots for numbers and hands. It is not important when the lower trapezoidal region is added, whether the hands are drawn before the number dots, or whether the window section is drawn before of after the upper circular region. Finally, the pendulum sprites are created and then animated within the clock window. The sequence determined by these authors is given below.

1. Shade large rectangle in orange
2. Shade window in white
3. Shade lower trapezoidal region in orange
4. Shade part of larger circle in orange
5. Continue with shading of circle, but in white
6. Display dots for numbers
7. Display hands
8. Draw initials
9. Draw lines to give the picture depth.
10. Create pendulum sprites
11. Animate pendulum sprites

To reflect this idea, procedures in CLOCK.SCENE are conceived. Enter the procedure defined below.

```
TO CLOCK.SCENE
READSHAPES "PENDULUMS
DOUBLECOLOR
FULLSCREEN
BG 5
```

```
CASE
FACE
TIME
HANDS
INITIALS
EMPHASIS
TICK.TOCK
END
```

Step 4. Now it is time to develop these procedures. Let's first construct the procedure for coloring a rectangular region. There are several schemes which could be developed. The one adopted for this example is to color the rectangle by repetitively drawing horizontal lines from the lower boundary of the rectangle to its upper boundary. Figure 6-8 illustrates the points of importance in formulating this procedure.

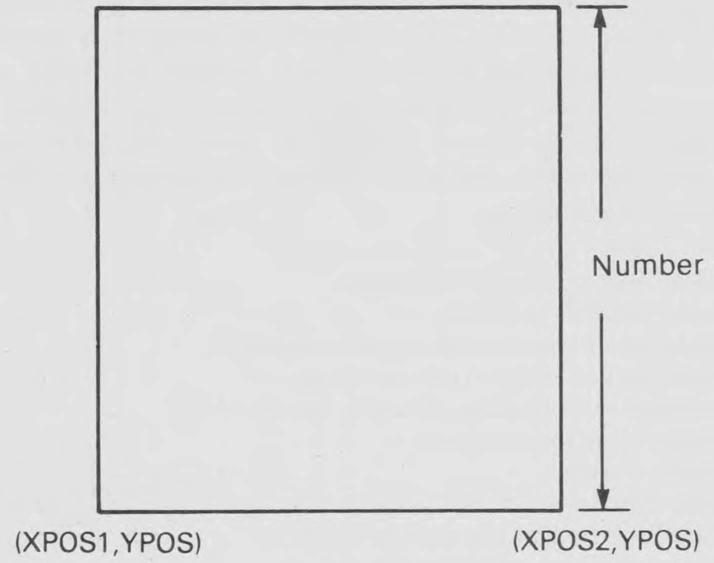


Figure 6-8.

The plan is to draw a horizontal line from XPOS1 to XPOS2 at row YPOS, then increase YPOS by 1 and continue to repeat this process. The number of repetitions is determined by the value of NUMBER.

RECTFILL defines the procedure that performs this process. Enter RECTFILL and test the procedure by entering the given input value. These are the actual values needed as one of the steps to create the clock. The value 110 is used since there are 110 (or 50 - (-60)) horizontal lines between the screen coordinates (30,50) and (30,-60). Refer to Figure 6-7 to verify these values. Similarly, 70 or (20 - (-50)) horizontal lines are used to shade the window, and 50 or (20 - (-30)) horizontal lines are used to shade the pendulum. Variable COLOR is used to control the changing of colors in the procedure.

```
TO RECTFILL :XPOS1 :XPOS2 :YPOS :COLOR :NUMBER
HT
IF :NUMBER = 0 STOP
PC :COLOR
PU
SETXY :XPOS1 :YPOS
PD
SETXY :XPOS2 :YPOS
RECTFILL :XPOS1 :XPOS2 :YPOS + 1 :COLOR :NUMBER - 1
END

TO WEDGEFILL :XPOS1 :XPOS2 :YPOS :COLOR :NUMBER
HT
IF :NUMBER = 0 STOP
PC :COLOR
PU SETXY :XPOS1 :YPOS PD
SETXY :XPOS2 :YPOS
WEDGEFILL :XPOS1 + 1 :XPOS2 - 1 :YPOS - 1 :COLOR
  :NUMBER - 1
END

TO CASE
RECTFILL -30 30 (-60) 8 110
RECTFILL -20 20 (-50) 1 70
WEDGEFILL -30 30 (-60) 8 10
PU HOME PD
END

CLOCK.SCENE
CS
```

The coloring of circles is achieved by repetitively drawing concentric circles, with each new circle having a radius of one less than the previous one. The CIRCLER procedure is used, but you must be careful to start the drawing at the correct point and at SETH 0. Coordinate (-40,70) in Figure 6-7 is used to begin coloring the larger circle.

The coloring needs only occur in orange ten times since the thickness of the circular frame is ten units. Then the same procedure can continue to draw concentric circles in white.

The COLORFILL procedure is given below. In this procedure, variable NUMBER controls the number of concentric circles to be drawn while COLOR sets the color. Read ARCS from the Utilities Disk and enter the procedure and test it by using the given values, which are the actual ones needed for the diagram.

```
READ "ARCS
TO CIRCLEFILL :RADIUS :COLOR :NUMBER
PC :COLOR
IF :NUMBER = 0 STOP
CIRCLER :RADIUS
PU SETH 90 FD 1 SETH 0 PD
CIRCLEFILL :RADIUS - 1 :COLOR :NUMBER - 1
END

TO FACE
PU FD 70 SETH 270 FD 40 SETH 0 PD
CIRCLEFILL 40 8 10
CIRCLEFILL 30 1 26
END

CLOCK.SCENE
```

The other procedures needed to complete the diagram are given below for you to study and enter. You might want to test each procedure as it is entered to observe the evolution of the clock.

```
TO TIME
HT
PC 0
PU SETXY 0 95 PD DOT
PU SETXY 22 70 PD DOT
PU SETXY 0 48 PD DOT
PU SETXY -22 70 PD DOT
END

TO DOT
REPEAT 4 [FD 1 RT 90]
END

TIME
```

```
TO HANDS
PC 0
PU SETXY 0 70 PD
SETH 0 FD 20
SETH 150 FD 4 BK 4
SETH 210 FD 4
PU SETXY 0 70 PD
SETH 60 FD 12
SETH 195 FD 4 BK 4
SETH 265 FD 4
END

HANDS

TO INITIALS
PC 0
PU SETXY -2 63 PD
SETXY -8 63
SETXY -8 60
SETXY -2 60
SETXY -2 57
SETXY -8 57
PU SETXY -2 63 PD
SETXY 8 63
SETXY 5 63
SETXY 5 57
END

INITIALS

TO EMPHASIS
PC 0
PU SETXY -30 (-60) PD
SETXY 30 -60
PU SETXY -27 (-63) PD
SETXY 27 (-63)
PU SETXY -24 (-66) PD
SETXY 24 (-66)
PU SETXY -42 70 SETH 0 PD
CIRCLER 42
END

EMPHASIS
```

Step 5. Now we shall create the three new sprites used to animate a pendulum within the window of the clock. First, read SPRED file from the Utility Disk. Use the sprite editor to modify sprites numbered 1, 2, and 3 to appear as illustrated in Figures 6-9, 6-10, and 6-11. Save these sprites in a file named PENDULUMS on your file disk. Enter the following procedures to position and animate the pendulum.

```
TO TICK.TOCK
EACH [1 2 3][BIGX BIGY PC 0]
EACH [1 2 3][FD 12 LT 90 FD 15]
EACH [2 3][HT]
REPEAT 10 [TELL 1 ST HT WAIT 2 TELL 2 ST WAIT 2 HT
  TELL 1 ST HT TELL 3 ST WAIT 2 HT]
TELL 1 ST
END
TO WAIT :DELAY
REPEAT :DELAY*300 []
END
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1											●	●	●											
2											●	●	●											
3											●	●	●											
4											●	●	●											
5											●	●	●											
6											●	●	●											
7											●	●	●											
8											●	●	●											
9											●	●	●											
10											●	●	●											
11											●	●	●											
12											●	●	●											
13											●	●	●											
14											●	●	●											
15											●	●	●											
16											●	●	●											
17										●	●	●	●	●										
18									●	●	●	●	●	●	●									
19									●	●	●	●	●	●	●									
20										●	●	●	●	●										
21											●	●	●											

Figure 6-9.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1											●	●	●											
2											●	●												
3										●	●	●												
4										●	●													
5									●	●	●													
6									●	●														
7								●	●	●														
8								●	●															
9							●	●	●															
10							●	●																
11						●	●	●																
12						●	●																	
13					●	●	●																	
14					●	●																		
15			●	●	●	●																		
16		●	●	●	●	●	●																	
17		●	●	●	●	●	●																	
18		●	●	●	●	●	●																	
19			●	●	●	●																		
20																								
21																								

Figure 6-10.

Step 6. The last step is to finally review all the tested procedures under one name. Study the sequence and input values of these commands and procedures used to create the diagram illustrated in Figure 6-7. Review the order of appearance of each clock part. Use the editor to modify CLOCK.SCENE, then execute the procedure to observe the final result, illustrated in Figure 6-12.

```
TO CLOCK.SCENE
READSHAPES "PENDULUMS
DOUBLECOLOR
FULLSCREEN
BG 5
CASE
FACE
TIME
HANDS
INITIALS
```

```
EMPHASIS
TICK.TOCK
PU HOME PD
END

CLOCK.SCENE

SAVE "TIME.PIECE
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1											●	●	●											
2												●	●											
3												●	●	●										
4													●	●										
5													●	●	●									
6														●	●									
7														●	●	●								
8															●	●								
9															●	●	●							
10																●	●							
11																●	●	●						
12																	●	●						
13																	●	●	●					
14																		●	●					
15																		●	●	●	●			
16																	●	●	●	●	●	●		
17																	●	●	●	●	●	●		
18																	●	●	●	●	●	●		
19																		●	●	●	●			
20																								
21																								

Figure 6-11.

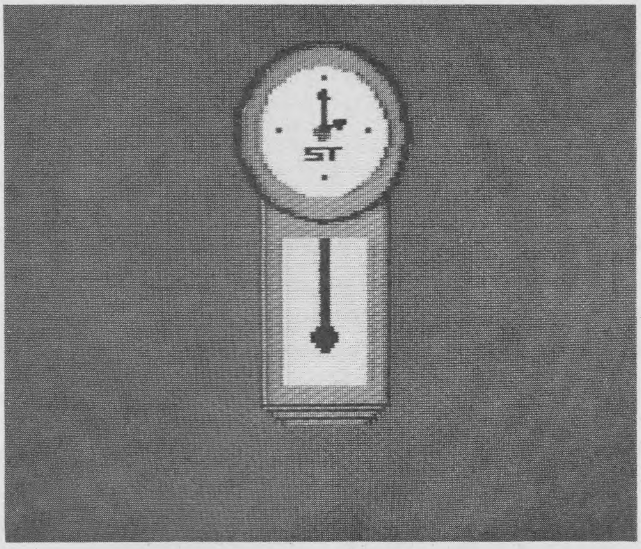


Figure 6-12.

Concluding Remarks

Congratulations! You have almost reached the end of this book. We hope you found the Logo activities a rewarding and motivating experience.

Logo is not a computer language one learns for the sake of learning a programming language. Instead, Logo is a vehicle by which problem solving and critical thinking take place. By mastering the fundamental concepts of Logo, you now have the opportunity to expand and disseminate its various applications in a learning environment.

First, if you are an educator, introduce or continue to use Logo in your classroom. Encourage your students to explore and elaborate on the activities you initiate. Get other teachers involved. Have the excitement of Logo permeate your entire school district.

Secondly, if you are a parent or child, Logo provides a medium where a family can learn together and share creative ideas at an academic level common to both groups. If you have read this book independently, introduce the remainder of your family to Logo.

Finally, if you are a computer hobbyist, one who simply enjoys working at a computer, you will want to suggest Logo to other hobbyists. Perhaps you will want to start a Logo group in your community where ideas and information are interchanged.

Learning with Logo is an ongoing process. Each interesting idea can serve as a catalyst for another. Your Logo experience should not end here. There are many challenges for you to accept in the Suggested Projects listed below. The Resource List also provides periodical articles, other books, and possible publications. Information from these will increase your depth and understanding of Logo beyond *COMMODORE LOGO: ACTIVITIES FOR EXPLORING TURTLE GRAPHICS.*

ENJOY EXPLORING YOUR THOUGHTS!

Suggested Projects

1. A circus wagon
2. A sunset at the beach
3. A floorplan
4. A spaceship
5. The solar system
6. A tennis court
7. A zoo
8. A sailboat regatta
9. A musical instrument
10. A campground
11. An original idea

Resource List

Books

- Abelson, Harold: LOGO for the Apple II. BYTE/McGraw-Hill, New York, 1983
- Bearden, Donna: 1, 2, 3 My Computer and Me. Reston Publishing Company, Reston, VA, 1983
- Bearden, Donna; Muller, James; Martin, Kathleen: The Turtle's Sourcebook. Reston Publishing Company, Reston, VA, 1983
- Bitter, Gary G.; Watson, Nancy: Apple Logo Primer. Reston Publishing Company, Reston, VA, 1983
- Grammer, Virginia; Goldenberg, Paul; Klotz, Leigh: LOGO—A Language For Learning. Commodore Business Machines Inc., West Chester, PA, 1983
- Moore, Margaret: LOGO Discoveries. Creative Publications, Palo Alto, CA, 1984
- Papert, Seymour: Mindstorms: Children, Computers, and Powerful Ideas. Basic Books, New York, 1980
- Papert, Seymour; diSessa, Andrea; Watt, Daniel; Weir, Sylvia: Final Report of the Brookline Logo Project. MIT Logo Group, Cambridge, MA, 1979
- Ross, Peter: Introducing Logo. Addison-Wesley Publishing Co., Reading, MA, 1983
- Sharp, Pamela: Turtlesteps. Robert J. Brady Co., Bowie, MD 1983
- Taylor, Robert P.: The Computer in the School: Tutor, Tool, Tutee. Teacher's College Press, New York, 1980

- Thornburg, David D.: Discovering Apple Logo: An Invitation to the Art and Pattern of Nature. Addison-Wesley Publishing Co., Reading, MA, 1983
- Thornburg, David D.: Picture This! Pilot Turtle Geometry, An Introduction to Computer Graphics For Kids of All Ages. Addison-Wesley Publishing Co., Reading, MA, 1982
- Watt, Daniel: Learning With Logo. BYTE/McGraw-Hill, New York, 1982

Periodical Articles

- Abelson, Harold: "A Beginner's Guide To Logo." BYTE, August 1982
- Bearden, Donna: "The Voice Of The Turtle." Softalk, October-November-December 1983
- Bearden, Donna; Muller, Jim: "Turtle Graphics: On and Off the Screen." Classroom Computer News, April 1983
- Billstein, Rick: "Learning Logo and Liking It." Computing Teacher, November 1982
- Bitter, Gary G.: "The Road To Computer Literacy" (a five part series). Electronic Learning, September-October-November 1982, January-February 1983
- Bowen, Barbara: "Using Logo in the Classroom: The Project Approach." Teaching and Computers, Spring 1983
- Carter, Richard: "Logo and the Great Debate." Microcomputing, September 1981
- Carter, Ricky: "The Beginner's Guide to Logo." Classroom Computer News, April 1983
- Cron, Mary: "Trouble in Logoland." Teaching, Learning, Computing, December 1983
- Faflick, Phillip: "Teaching the Turtle New Tricks." Time, October 11, 1982
- Harvey, Brian: "Why Logo?" BYTE, August 1982

- Lawler, Robert: "Designing Computer-Based Microworlds." BYTE, August 1982
- Lough, Tom: "Exploring New Horizons with Logo: How One School is Charting the Course. Part 2." Electronic Learning, April 1983
- Lough, Tom: "Logo: Discovery Learning with the Classroom's Newest Pet." Electronic Learning, March 1983
- Lough, Tom; Tipps, Steve: "Logo Notebook." Teaching and Computers, September 1983
- Lough, Tom; Tipps, Steve: "Is There Logo After Turtle Graphics?" Classroom Computer News, April 1983
- Markuson, Carolyn; Tobias, Joyce; Lough, Tom: "Logo Fever: The Computer Language Every School is Catching." Instructor, January 1983
- Moore, Mary Jo: "The Art of Teaching Logo." Classroom Computer News, April 1983
- Muller, Jim: "The Friendly Computer Language." Creative Computing, October 1982
- Nelson, Harold: "Logo: Not Just For Kids." Microcomputing, March 1982
- Overall, Theresa et al: "Learning with Logo at the Lamplighter School." Microcomputing, September 1981
- Papert, Seymour: "New Cultures from New Technologies." BYTE, September 1980
- Piele, Donald T.: "Beyond Turtle Graphics." Creative Computing, March 1983
- Riordon, Tim: "Creating a Logo Environment." Computing Teacher, November 1982
- Rosser, Phyllis: "Computers in the Classroom: How They May Change the Way Children Learn." Teachers and Writers Magazine, March-April 1983
- Rousseau, Joseph; Smith, Stephen: "Wither Goes The Turtle?" Microcomputing, September 1981
- Soloman, Cynthia: "Introducing Logo to Children." BYTE, August 1982
- Upitis, Rena: "Logo and the Primary-Junior Pupil: One Student's First Encounter." Computing Teacher, November 1982

- Watt, Daniel: "A Comparison of the Problem Solving Styles of Two Children Learning Logo." Proceedings of the National Computing Conference, 1979
- Watt, Daniel: "Logo in the Schools." BYTE, August 1982
- Watt, Daniel: "Teaching Turtles: Logo as an Environment for Learning." Popular Computing, July 1982
- Watt, Molly: "What is Logo?" Creative Computing, October 1982
- Weinreb, William: "Problem Solving with Logo." BYTE, November 1982
- Weir, Sylvia: "Logo and the Exceptional Child." Microcomputing, September 1981
- Williams, Greg: "Logo for the Apple II, the TI-99/4A, and the TRS-80 Color Computer." BYTE, August 1982

Subscriptions

TurtleTalk (four issues per year)
TurtleTalk
955 Greenwood Blvd.
Issaquah, WA 98027

Turtle News (six issues per year)
Young People's Logo Association
P.O. Box 855067
Richardson, TX 75085

Answers To Turtle Testers

Chapter 1

Answer to Activity 1.1 Tester (page 7)

```
DRAW
HT
FD 120
CS
ST
BK 75
```

Answer to Activity 1.2A Tester (page 13)

```
DRAW
FD 60
RT 90
FD 40
RT 90
FD 60
RT 90
FD 40
RT 90
```

Answer to Activity 1.2B Tester (page 13)

```
DRAW
FD 40
LT 90
FD 80
LT 90
FD 40
LT 90
FD 80
LT 90
```

Answer to Activity 1.3 Tester (page 18)

```
DRAW
BK 50
FD 100
RT 90
FD 50
RT 90
FD 45
RT 45
FD 10
SETH 270
FD 43
BK 43
SETH 135
FD 10
RT 45
FD 45
RT 90
FD 50
RT 90
```

Answer to Activity 1.4 Tester (page 24)

```
DRAW
PU
FD 30
PD
REPEAT 6 [FD 40 LT 360/6]
REPEAT 6 [FD 40 RT 360/6]
RT 120
FD 40
RT 120
FD 40
SETH 0
BK 40
REPEAT 6 [FD 40 LT 60]
REPEAT 6 [FD 40 RT 60]
PU HOME PD
```

Chapter 2

Answer to Activity 2.1 Tester (page 49)

```
TO PENTAGON
```

```
REPEAT 5 [FD 40 RT 360/5]
END

TO TESTER2.1
REPEAT 6 [PENTAGON RT 360/6]
END

TESTER 2.1
```

Answer To Activity 2.2 Tester (page 57)

```
TO TRIANGLE
REPEAT 3 [FD 35 RT 360/3]
END

TO DECAGON
REPEAT 10 [FD 35 RT 360/10]
END

TO NONAGON
REPEAT 9 [FD 35 RT 360/9]
END

TO OCTAGON
REPEAT 8 [FD 35 RT 360/8]
END

TO HEPTAGON
REPEAT 7 [FD 35 RT 360/7]
END

TO HEXAGON
REPEAT 6 [FD 35 RT 360/6]
END

TO PENTAGON
REPEAT 5 [FD 35 RT 360/5]
END

SAVE "POLYGONS
GOODBYE
CATALOG
```

Chapter 3

Answer to Activity 3.1 Tester (page 107)

```
GOODBYE
TO SQUARE :SIDE
REPEAT 4 [FD :SIDE RT 90]
END

TO SQUARES.2
SQUARE 10
SQUARE 30
SQUARE 50
SQUARE 70
HT
END

TO SQDESIGN.2 :MANY
REPEAT :MANY [SQUARES.2 RT 360/:MANY]
END

SQDESIGN.2 40
```

Answer to Activity 3.6 Tester (page 156)

```
TO CIRCLES :RADIUS
REPEAT 5 [CIRCLER :RADIUS RT 72]
END

TO MANY.CIRCLES :RADIUS :INCREMENT
NOWRAP
CIRCLES :RADIUS
MANY.CIRCLES :RADIUS + :INCREMENT :INCREMENT
END

MANY.CIRCLES 30 10
WRAP
```

Answer to Activity 3.7 Tester (page 171)

```
TO FILL :SIDE :INCREMENT :MANY
IF :SIDE < :INCREMENT STOP
REPEAT :MANY [FD :SIDE LT 360/:MANY]
FILL :SIDE - :INCREMENT :INCREMENT :MANY
END
```

```
TO COLORFILL :SIDE :INCREMENT :MANY :COLOR
PC :COLOR
FILL :SIDE :INCREMENT :MANY
END

TO ARRANGE
FULLSCREEN
MOVE 270 60
MOVE 0 40
COLORFILL 50 .5 5 4
MOVE 90 200
COLORFILL 25 .4 8 8
MOVE 180 120
COLORFILL 20 .4 6 6
MOVE 270 200
COLORFILL 25 .3 10 5
PU HOME PD
END

TO MOVE :ANGLE :MORE
PU SETH :ANGLE FD :MORE SETH 0 PD
END

ARRANGE
```

Answer to Activity 3.8 Tester (page 179)

```
TO GYROTRI :SIDE
HT
RT 30
TRIANGLE :SIDE
END

TO TRIANGLE :SIDE
IF :SIDE < 4 STOP
REPEAT 3 [FD :SIDE RT 120]
FD :SIDE/2 RT 60
TRIANGLE :SIDE/2
END

GYROTRI 120
```

Chapter 4

Answer to Activity 4.1 Tester (page 199)

```
TO LONG :YPOSITION
```

```
PU SETX 0
SETY :YPOSITION
PD FD :YPOSITION * -2
END

TO LAT :XPOSITION
PU
SETX :XPOSITION
SETY 0
SETH 90
PD FD :XPOSITION * -2
END

TO NEGATIVE :SIDE
PD
SETH 90
FD :SIDE * 2
END

TO POSITIVE :SIDE
PD REPEAT 4 [FD :SIDE BK :SIDE RT 90]
END

TO RBRACKET :SIDE :XPOSITION :YPOSITION :ANGLE
MOVE :XPOSITION :YPOSITION :ANGLE
FD :SIDE
SETH 0
FD :SIDE * 2
SETH 270
FD :SIDE
END

TO LBRACKET :SIDE :XPOSITION :YPOSITION :ANGLE
MOVE :XPOSITION :YPOSITION :ANGLE
FD :SIDE
SETH 0
FD :SIDE * 2
SETH 90
FD :SIDE
END

TO AXIS :XPOSITION :YPOSITION
HT
```

```
LONG :YPOSITION
LAT :XPOSITION
PU SETXY 10 (-10) STAMP [(0,0)]
SETXY 0 (-1 * :YPOSITION + 8) STAMPCHAR "Y
SETXY (-1 * :XPOSITION + 5) 0 STAMPCHAR "X
END

TO GRID.LABELS
FULLSCREEN
AXIS -149 (-120)
LBRACKET 10 30 40 270
RBRACKET 10 80 40 90
MOVE 40 50 0
POSITIVE 10
MOVE 65 50 0
POSITIVE 10
LBRACKET 10 30 (-40) 270
RBRACKET 10 80 (-40) 90
MOVE 40 (-30) 0
POSITIVE 10
MOVE 65 (-30) 0
NEGATIVE 10
LBRACKET 10 (-80) 40 270
RBRACKET 10 (-30) 40 90
MOVE -80 50 0
NEGATIVE 10
MOVE -40 50 0
POSITIVE 10
LBRACKET 10 (-80) (-40) 270
RBRACKET 10 (-20) (-40) 90
MOVE -80 (-30) 0
NEGATIVE 10
MOVE -40 (-30) 0
NEGATIVE 10
HT
END

TO MOVE :XPOSITION :YPOSITION :ANGLE
PU SETPOS LIST :XPOSITION :YPOSITION
SETH :ANGLE
PD
END

GRID.LABELS
```

Answer to Activity 4.2 Tester (page 207)

```
TO BOW
PC 1
RT 45
REPEAT 3 [FD 8 RT 120]
LT 90
REPEAT 3 [FD 8 LT 120]
HT
END

TO FRAME :XPOS :YPOS :COLORFRAME
PU
SETXY :XPOS :YPOS
PD PC :COLORFRAME
SETXY :XPOS :YPOS + 50
SETXY :XPOS - 20 :YPOS + 30
SETXY :XPOS :YPOS
SETXY :XPOS + 20 :YPOS + 30
SETXY :XPOS :YPOS + 50
SETXY :XPOS + 20 :YPOS + 30
SETXY :XPOS - 20 :YPOS + 30
PU
SETXY :XPOS :YPOS
PC 1
PD
END

TO KITE :XPOS :YPOS :COLORFRAME :COLORTAIL
FRAME :XPOS :YPOS :COLORFRAME
TAIL :XPOS :YPOS :COLORTAIL
PU HOME PD
END

TO MANY.KITES
FULLSCREEN
KITE 0 0 4 1
KITE -50 10 9 5
KITE 50 (-10) 1 6
END

TO TAIL :XPOS :YPOS :COLORTAIL
PC :COLORTAIL
SETH TOWARDS :XPOS - 10 :YPOS - 10
```

```
FD 16
SETH TOWARDS :XPOS - 16 :YPOS - 20
FD 8
SETH TOWARDS XPOS - 14 :YPOS - 20
FD 8
BOW
PC :COLORTAIL
SETH TOWARDS :XPOS - 7 :YPOS - 27
FD 16
SETH TOWARDS :XPOS + 24 :YPOS - 49
FD 16
BK 10
BOW
END

MANY.KITES
```

Answer to Activity 4.3 Tester (page 214)

```
TO STAR.TESS
ACROSS 40 (-115) (-110) 8
HT
END

TO SIDE :DISTANCE
RT 45 FD :DISTANCE
LT 45 FD :DISTANCE
LT 135 FD :DISTANCE
LT 45 FD :DISTANCE
RT 180
END

TO VARICUBE :DISTANCE
BOX :DISTANCE
SIDE :DISTANCE
RT 90
FD :DISTANCE
LT 90
SIDE :DISTANCE
LT 90
FD :DISTANCE
RT 135
FD :DISTANCE
LT 45
```

```
BOX :DISTANCE
RT 45
BK :DISTANCE
LT 45
END

TO COLOR
PC RANDOM 11
END

TO BOX :DISTANCE
REPEAT 4 [FD :DISTANCE RT 90]
END

TO ACROSS :LENGTH :XPLACE :YPLACE :DISTANCE
IF :XPLACE > 159 - :LENGTH STOP
VROW :LENGTH :XPLACE :YPLACE :DISTANCE
MOVE :LENGTH
ACROSS :LENGTH :XPLACE + :LENGTH :YPLACE :DISTANCE
END

TO STAR :DISTANCE :MANY
COLOR
REPEAT :MANY [VARICUBE :DISTANCE RT 360/:MANY]
PR POS
END

TO VROW :LENGTH :XPLACE :YPLACE :DISTANCE
FULLSCREEN
PU SETXY :XPLACE :YPLACE PD
MAKE "POSX :XPLACE
MAKE "POSY :YPLACE
MAKE "NUMBER (2 * :YPLACE)/:LENGTH
IF :NUMBER < 0 MAKE "NUMBER :NUMBER * (-1)
MAKE "NUMBER INTEGER :NUMBER
REPEAT :NUMBER [STAR :DISTANCE :MANY MOVE :LENGTH]
PU SETXY :POSX :POSY PD
END

TO MOVE :LENGTH
PU FD :LENGTH PD
END

STAR.TESS
```

Chapter 5

Answer to Activity 5.1 Tester (page 232)

```
TO CRAWL :N
RT 90
REPEAT :N [BK 1 BK RANDOM 26]
LT 90
END

TO HOP :N
RT 90
REPEAT :N [BK 10 LT 90 BK 10 RT 90 BK 3 RT 90 BK
  10 LT 90]
LT 90
END

TO GREAT.RACE
TELL 0 RT 90 FD 318 HT
TELL 5 SETXY 130 100 ST
TELL 6 SETXY 130 20 ST
TELL 4 SETXY 130 (-40) ST
REPEAT 10 [TELL 6 HOP 1 TELL 5 CRAWL 3 TELL 4
  CRAWL 2]
END

GREAT.RACE
SAVE "COMPETITION
```

Answer to Activity 5.2 Tester (page 241)

```
TO TILT
REMOVE.SHAPES 0
OUTLINE
TELL 3 PC 1 ST SETXY 60 (-68) SETH 0
OPENGATE
FD 300
TRAPLOOP
END

TO REMOVE.SHAPES :NUMBER
IF :NUMBER = 8 STOP
TELL :NUMBER HOME HT
REMOVE.SHAPES :NUMBER + 1
```

```
CS
CLEARTEXT
END

TO OUTLINE
ASK 0 [PU PC 1 SETXY 70 (-60) PD SETH 0]
ASK 0 [FD 120 ARCL 60 180 FD 120 LT 90 FD 120 PC
  -1 BK 15 PC 1 LT 90 FD 80]
ASK 0 [SETH 45 FD 20]
TELL 2 PC 2 ST SETXY 35 50
TELL 4 PC 7 ST SETXY -20 80
TELL 5 PC 13 ST SETXY -20 (-30)
TELL 6 PC 6 ST SETY 100
TELL 7 PC 4 ST SETXY -10 30
ASK 0 [PU SETXY 70 60 PD SETH 330 FD 60]
ASK 0 [HT]
END

TO OPENGATE
ASK 0 [PC -1 PU SETXY 50 20 PD SETH 45 FD 20 PC
  1]
END

TO TRAPLOOP
IF RC? STOP
IF TB? COLORBALL BK 3 RT RANDOM 180 CLOSEGATE
IF TS? COLORBUMPER 0 BK 3 LT RANDOM 180
TELL 3
FD 1
TRAPLOOP
END

TO COLORBALL
TELL 3 PC RANDOM 11
END

TO COLORBUMPER :MANY
IF :MANY = 5 STOP
COLOR.OFF
WAIT 1
COLOR.ON
WAIT 1
COLORBUMPER :MANY + 1
END
```

```
TO COLOR.ON
EACH [2 4 5 6 7] [PC RANDOM 11]
END

TO COLOR.OFF
EACH [2 4 5 6 7] [PC 11]
END

TO WAIT :TIME
REPEAT 100 * :TIME []
END

TO CLOSEGATE
ASK 0 [PU SETXY 55 20 PD SETH 45 FD 20]
END

TILT
SAVE "GAME1
```

Answers to Activity 5.3 Tester (page 245)

```
TO BONZAI
SETUP
TAKEOFF
DROP
HOMEWARD
EACH [5 7] [HT]
END

TO SETUP
CLEARTEXT
EACH [0 5 7] [HOME HT]
TELL 5 PC 2 BIGX BIGY SETXY 130 100 LT 90 ST
TELL 7 PC 7 SMALLX SMALLY SETY 100 HT
END

TO TAKEOFF
TELL 5
IF XCOR = 0 STOP
FD 2 WAIT 2
TAKEOFF
END

TO DROP
TELL 7 BK 10 ST
```

```
IF YCOR < 35 TELL 7 BIGX
IF YCOR < 1 TELL 7 BIGY
IF YCOR < -60 STOP
DROP
END

TO HOMEWARD
TELL 5
IF XCOR < -50 SMALLX
IF XCOR < -100 SMALLY
IF XCOR < -130 STOP
FD 2 WAIT 2
HOMEWARD
END

BONZAI
SAVE "BOMBARDMENT
```

INDEX

angle, 8, 9
arc, 32
ARCL, 32-35, 37
ARCR, 33-35, 37
ARCS, 26, 29, 34
arithmetic operations, 181
ASK, 233-238

BACK(BK), 3-6, 36
BACKGROUND (BG), 114-122, 180
BIGX, 242, 243, 255
BIGY, 243, 255
brackets[ ], 20-22

Cartesian coordinates, 189, 196
CATALOG, 25, 26, 42, 58, 59, 61, 62, 83
circle, 25, 29, 30
CIRCLEL, 26-29, 34, 37
CIRCLER, 26-30, 37
CLEARSCREEN(CS), 15, 17, 37
CLEARTEXT, 60, 62, 181
CLR, 94, 248, 255
CO, 135-138, 181
cold boot, 2
color chart, 112
constants, 103
CONTROL(CTRL)key, 25, 26, 64, 65, 78, 83-90, 93, 94
correcting typing errors, 2
CRAWL, 228, 231, 232, 258
CRSR keys, 65-67, 89, 90, 94, 248
CTRL-A, 66-68, 89, 94
CTRL-B, 66, 67, 94
CTRL-C, 43, 48, 65, 76, 90, 94
CTRL-D, 67, 68, 90, 94
CTRL-F, 66, 67, 94
CTRL-G, 43, 65, 90, 94
CTRL-K, 66-68, 90, 94
CTRL-L, 66-68, 89, 94
CTRL-N, 65, 68, 90, 94
CTRL-O, 65-68, 72, 90, 94
CTRL-P, 65-68, 72, 90, 94
CTRL-W, 26, 62, 94
CTRL-Z, 94
cursor, 1

DOUBLECOLOR mode, 110-112, 171, 181
DRAW, 2, 5, 36
DRAWSTATE, 137, 138

EACH, 227, 230
EDIT, 65, 66, 68, 93-95
edit buffer, 64, 65
edit mode, 42, 45, 89, 90
EDSH, 247-251
ER, 68, 69
ERASEFILE, 61, 62
ERASE ALL, 61, 62
exterior angle, 21, 22

f1, 6, 51, 80, 90, 94
f3, 5, 6, 80, 90, 94
f5, 6, 80, 90, 94
format, 41, 42, 45
FORWARD(FD), 3-5
FULLSCREEN (f5), 4, 6
function keys, 4-6

global variable, 125, 131, 160, 169
GOODBYE, 53, 54, 59, 62, 230

HEADING, 7, 8, 11, 12
HIDETURTLE(HT), 3, 5, 36
HOME, 11, 12, 17, 37, 94, 248, 255
HOP, 228, 231, 232, 258

IF (conditional expression), 182, 158-162
input, 1
INST/DEL key, 2, 89, 95

LEFT(LT), 8, 9, 11, 36
LOAD, 1, 2
local variables, 125, 131

MAKE, 128-132, 182

NODRAW, 59, 62
NOWRAP, 145, 147, 148, 154

ordered pair (X,Y), 189, 196

Part-to-Whole Programming, 264-271
PENCOLOR (PC), 109, 111, 180
PENDOWN(PD), 14, 16, 17, 37
PENERASE(PC-1), 15, 17, 37, 125
PENUP(PU), 14, 16, 17, 37
pixel, 217, 247, 251
PO, 60, 62
PO ALL, 60, 62
polygon, 21
POTS, 59-62
predicates and functions (<,>,=), 158, 160, 163, 181
primitives, 42
PRINT(PR), 11, 37, 181
PR HEADING, 11, 12, 37
procedures, 45, 54, 122
prompt, 2
PR XCOR, 190-196, 218
PR YCOR, 190-196, 218

RANDOM, 130-132, 181
READ, 26, 59-62
READSHAPES, 251, 253
recursion (complex), 172-177
recursion (simple), 144-154, 158-163, 176, 177
REPEAT, 20-22, 37
RESTORE, 42
RIGHT (RT), 8-12, 36
RUN/STOP key, 42

SAVE, 53, 54
SAVESHAPES, 250, 251, 253
Screen (Cartesian) coordinates, 189, 196, 217, 219
SENTENCE (SE), 150, 151, 154
SETH, 10, 12, 36
SETH TOWARDS, 203-205, 218, 219
SETX, 191-196, 218
SETXY, 192-197, 217
SETY, 191-196, 218
SHOWTURTLE (ST), 3, 5, 37
SINGLECOLOR mode, 111, 112, 171, 181
SMALLX, 243, 255
SMALLY, 243, 255
SPLITSCREEN (f3), 4-6, 36
SPRED, 246-251
SPRITEDEMOS, 228, 231, 238, 243, 256
Sprite Editor, 246, 251, 255
Sprites, 225, 229, 230, 238, 239, 251
SPRITES, 225, 229, 230, 244, 256
Sprite Shape Files, 230
STAMP, 194, 196, 197, 219
STAMPER, 194, 196

STAMPCHAR, 194, 196, 197, 219
STOP, 158-163
syntax, 263

TEACH file, 76-86, 89-95
TELL, 225-230
tessellations, 207-217, 219-223
TEXTBG, 136-138
TEXTCOLOR, 136-138
TEXTSCREEN (f1), 4-6, 80, 90, 94
TO, 43-45, 89
toplevel mode, 45, 77, 87, 93-95
Total Turtle Trip Theorem, 22, 34
TRAPLOOP, 236-240

Utilities Disk, 76, 80, 229, 230, 256

variable, 97-104, 111, 182

WHO, 233, 238
Whole-to-Part Programming, 264, 271-281
workspace, 54
WRAP, 145, 148, 153, 181

XCOR, 189-196
YCOR, 189-196

COMMODORE LOGO
Activities For Exploring Turtle Graphics

LOGO IN A TURTLESHELL

FUNDAMENTAL COMMANDS

ST	ShowTurtle
HT	HideTurtle
FD __	ForwarD __ steps
BK __	BacK __ steps
RT __	RighT __ degrees
LT __	LefT __ degrees
CS	ClearScreen
PU	PenUp
PD	PenDown
PC –1	Prepares to erase
PC 1	Restores drawing state

POSITIONS/DIRECTIONS

SETH __	SETHeading to __
SETXY __ __	Sends turtle to coordinate (__ __)
SETX __	Sends turtle to X position __
SETY __	Sends turtle to Y position __
SETH TOWARDS __ __	Aims turtle toward coordinate (__ __)
HOME	Sends turtle to center screen
DRAW	Prepares turtle to draw
PR HEADING	Prints value of current SETH direction
PR YCOR	Prints value of Y position
PR XCOR	Prints value of X position

PROCEDURE DEFINITION

TO ____	Begins a definition for the procedure ____
TO __:__	Begins a definition for the procedure ____ containing a variable :____

COLOR COMMANDS

PC ____	PenColor is ____
BG ____	BackGround color is ____
SINGLECOLOR	Thinnest line
DOUBLECOLOR	Thickest line

COLOR CHART

0	Black	**8**	Orange
1	White	**9**	Brown
2	Red	**10**	Lt Red
3	Cyan	**11**	Gray 1
4	Purple	**12**	Gray 2
5	Green	**13**	Lt Green
6	Blue	**14**	Lt Blue
7	Yellow	**15**	Gray 3

SCREEN DISPLAY

f1	Textscreen
f3	Splitscreen
f5	Fullscreen

CIRCLES/ARCS

(Read in from Utility Disk file ARCS)

CIRCLER ____	Circle to right with a radius of ____
CIRCLEL ____	Circle to left with a radius of ____
ARCR __ __	Arc to right with radius ____ and degrees ____
ARCL __ __	Arc to left with radius ____ and degrees ____

FILE MANIPULATION

READ "____	Place the file ____ in workspace
CATALOG	Displays all file names on disk
SAVE "____	Transfers all procedures in workspace to file disk under file name ____
GOODBYE	Erases all procedures in workspace
POTS	Prints Out all TitleS
ERASE ____	Erases a procedure from workspace
ERASE [__ __]	Erases all procedures enclosed within brackets

PO ____	Prints procedure ____
PO [__ __ __]	Prints all procedures enclosed within brackets
ERASEFILE ____	Removes file ____ from a file disk

SPRITE COMMANDS

(Read in from Utility Disk file SPRITES)

TELL __	Addresses the sprite __
WHO	Responds with number of current sprite
ASK __ [__ __]	Tells sprite __ to follow directions enclosed in brackets
BIGX	Doubles sprite width
BIGY	Doubles sprite height
SMALLX	Original sprite width
SMALLY	Original sprite height
SETSHAPE __	Commands current sprite to assume sprite shape __

EDITING COMMANDS

EDIT ____	Places procedure ____ in the edit buffer
→	Moves ■ right
←	Moves ■ left
↑	Moves ■ up
↓	Moves ■ down
CTRL-A	Moves ■ to beginning of line
CTRL-B	Moves ■ one screenful backward
CTRL-C	Exits editor and defines procedure
CTRL-D	Deletes character under ■
CTRL-F	Moves ■ one screenful forward
CTRL-G	Exits editor and aborts procedure
CTRL-K	Deletes characters right of ■
CTRL-L	Moves ■ to end of line
CTRL-N	Moves ■ to next line
CTRL-O	Opens a new line at ■
CTRL-P	Moves ■ to previous line

SPRITE EDITING COMMANDS

(Read in from Utility Disk file SPRED)

EDSH	Places current sprite in editor
+	Fills in pixel under ■
*	Fills in pixel under ■ and moves to the right
−	Erases pixel under ■
SPACE	Erases pixel under ■ and moves to the right
DEL	Moves to left and erases pixel under ■
↓	Moves ■ down one line
↑	Moves ■ up one line
→	Moves ■ to right one pixel
←	Moves ■ to left one pixel
RETURN	Moves ■ to beginning of next line
CLR	Moves ■ to upper left corner and erases all pixels
HOME	Moves ■ to upper left corner
X	Displays opposite shape value (BIGY/SMALLY)
Y	Displays opposite shape value (BIGY/SMALLY)
CTRL-9	Displays sprite inverse
CTRL-C	Defines sprite shape and leaves editor

Parents, Teachers! Here's An Exciting And Creative New Way
To Explore Turtle Graphics Together With Your Children And Students!

Acclaim from prepublication reviewers!

". . .extremely well-organized. . .the authors truly take the reader by hand and walk him along the turtle path!"

"This book fills a tremendous gap in the current literature on LOGO for the Commodore 64. . .the organization and content are by far the best I've seen in any teaching guide!"

COMMODORE LOGO: ACTIVITIES FOR EXPLORING TURTLE GRAPHICS

Harold Bailey, Trudy H. Doran, and Kathleen Brautigam

A "hands-on" activities guide that offers children an exciting and creative approach to exploring turtle graphics! This guide includes a series of "hands-on" activities that begins with basic turtle commands and concepts. . .then expands to more sophisticated graphics design. What's more, this book goes beyond the "discovery-based" approach to learning LOGO—inside this book you'll undergo the "guided discovery" approach that allows youngsters the ability to grasp powerful ideas through systematic planning and problem-solving techniques!

Divided into six sequentially-designed chapters, **Commodore LOGO: Activities for Exploring Turtle Graphics** shows users how to control movement of the "turtle" through simple LOGO commands. . .how to combine these commands to form procedures. . .how to create and animate sprites. . .how to use color. . .how to use coordinates for effective graphics design. . .and how to put it all together to produce complex screen displays! Each activity presented within a chapter is divided into five concise "turtle" sections as well, offering users immediate and practical feedback from the initially given LOGO command to the final activity evaluation!

CONTENTS

Exploring Turtle Territories • Preserving Turtle Tracks • Extending Turtle Trips • Coordinating Turtle Points • Recruiting The Turtle's Team • Creating Turtle Temptations • Selected Answers To Turtle Testers

0 08930 33766

ISBN 0-89303-376-6